МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
Вычислительной техники	Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

УНИВЕРСАЛЬНЫЙ КЛЮЧ ИДЕНТИФИКАЦИИ И КОНТРОЛЯ RFID-МЕТОК НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА STM32

Студент 31.05.22 Амплеев Д.О.
(подпись, дата)

Руководитель 31.05.22 Бычков А.С.
(подпись, дата)

Нормоконтролёр 31.05.22 Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой М.А. Митрохин
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 21.06.22 № 21)

Секретарь ГЭК Гурин Е.И.
(подпись)

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«26» 04 2022 г.

ЗАДАНИЕ

НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Амплеев Даниил Олегович гр.18ВВ1 факультета вычислительной техники.
2. Тема работы: «Универсальный ключ идентификации и контроля RFID-меток на основе микрс контроллера STM32»

Тема утверждена приказом ПГУ № 357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ Бычков А.С.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать «Универсальный ключ идентификации и контроля RFID-меток на основе микроконтроллера STM32», выполняющий копирование, хранение и повторение RFID-ключ. Устройство должно иметь малые габариты и быть оборудовано OLED-дисплеем для отображения меню.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих систем эмуляции RFID-меток, изучить способ взаимодействия микроконтроллера STM32 с OLED-дисплеем по I2C, изучить среду программирования микроконтроллера STM32 «STM32CubeIDE».

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2022-30.04.2022	
2.	Анализ требований	10	01.05.2022-02.05.2022	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2022-07.05.2022	
4.	Описание алгоритмов	40	08.05.2022-11.05.2022	
5.	Разработка универсального ключа идентификации и контроля RFID-меток на основе микроконтроллера STM32	150	12.05.2022-29.05.2022	
6.	Оформление пояснительной записки	24	30.05.2022-31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы 26.04.22 Бычков А.С.

(подпись, дата)

Задание к исполнению принял студент 26.04.22 Амплеев Д.О.

(подпись, дата)

Работу к защите допустить

Декан факультета 17.06.22 Фионова Л.Р.

(подпись, дата)

Реферат

Бакалаврская работа содержит 99 листов, 25 рисунков, 2 таблицы, 5 источников и 3 приложения.

РАЗРАБОТКА УНИВЕРСАЛЬНОГО КЛЮЧА ИДЕНТИФИКАЦИИ И КОНТРОЛЯ *RFID*-МЕТОК НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА *STM32*.

Цель работы – разработать универсальный ключ идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*.

Объектами разработки являются:

– универсальный ключ идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*;

– программное обеспечение устройства.

В результате проделанной работы была разработана схема электрическая принципиальная и спроектировано устройство контроля и идентификации *RFID*-меток. Так же было разработано ПО для работы устройства.

Содержание

Перечень сокращений и обозначений 5
Введение 7
1 Назначение, состав, область применения и принципы функционирования универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32* 8
1.1 Обзор используемого оборудования. 10
1.1.1 Микроконтроллер *STM32F030* 10
1.1.2 Операционный усилитель *NE532D* 13
1.1.3 Стабилизатор напряжения *LM3940IMPX* 14
1.2 Принцип функционирования *RFID* 15
1.3 Принципы функционирования универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*. 17
1.4 Разработка схемы электирической структурной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM*32 18
1.5 Выбор среды разработки 20
1.6 Разработка схемы электрической принципиальной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*. 20
1.6.1 Выбор элементной базы 22
1.6.2 Создание библиотеки УГО. 31
1.6.3 Описание схемы электрической принципиальной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM*32 34
1.6.4 Расчётно - пояснительная часть. 35
1.6.5 Расчет блока управления *RFID*. 36
1.7 Разработка программного обеспечения 37

1.7.1 Выбор среды разработки ПО 37
1.7.2 Написание программного обеспечения. 40
1.7.3 Принцип работы *I2C*. 41
1.7.4 Структура ключа *RFID*. 43
2 Охрана труда 45
2.1 Нормы освещенности рабочей зоны 45
2.1.1 Нормы освещенности рабочей зоны 45
2.1.2 Нормы освещённости при работе за компьютером. 45
2.2 Правила безопасности при работе с ВТ 45
2.2.1 Меры безопасности перед началом работы за ВТ. 45
2.2.2 Техника безопасности во время работы за ВТ. 45
2.2.3 Техника безопасности по окончанию работы за ВТ 46
2.3 Правила безопасности при монтаже радиоэлементов. 46
2.3.1 Нормы освещенности рабочей зоны 46
2.3.2 Меры безопасности во время монтажа. 46
2.3.3 Меры безопасности после завершения монтажа. 47
Заключение 48
Список использованных источников 49
Приложение А1 - Схема электрическая структурная 50
Приложение А2 - Схема электрическая принципиальная. 52
Приложение А3 - Схема программы. 54
Приложение Б – Листинг программы. 57
Приложение В – Презентация. 96

Перечень сокращений и обозначений

ARM - от англ. *Advanced RISC Machine* — усовершенствованная RISC-машина *Acorn RISC Machine* — система команд и семейство описаний и готовых топологий 32-битных и 64-битных микропроцессорных/ микроконтроллерных ядер, разрабатываемых компанией *ARM Limited*

CRC – техника контрольной суммы

I2C – последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов

*MOSFET (*англ. *metal-oxide-semiconductor field effect transistor)* - полупроводниковая структура, применяемая при производстве микросхем и дискретных полевых транзисторов. Полупроводниковые приборы на основе этой структуры называют МОП-транзисторами (аббревиатура от слов «металл-оксид-полупроводник», англ. *metal-oxide-semiconductor field effect transistor*, сокращенно «*MOSFET*»), МДП-транзисторами (от слов «металл-диэлектрик-полупроводник») или транзисторами с изолированным затвором (так как у таких транзисторов затвор отделён от канала тонким слоем диэлектрика)

*OLED (*англ. *organic light-emitting diode)*- полупроводниковый прибор, изготовленный из органических соединений, эффективно излучающих свет при прохождении через них электрического тока

QR (англ. *Quick Response code* — код быстрого отклика; сокр. *QR code*) — тип матричных штриховых кодов (или двухмерных штриховых кодов)

RFID (англ. *Radio Frequency IDentification*, радиочастотная идентификация) - способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или *RFID*-метках

RTC – часы реального времени

RC – в данном случае является обозначением *RC*-генераторов которые генерируют синусоидальный выходной сигнал, состоят из усилителя и частотно-

избирательного элемента — фильтра. *RC*-генератора, это генератор основанный на базе *RC*-цепей

USART (англ. *Universal Asynchronous Receiver-Transmitter)* - узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами

Транспондер - *RFID*-метка

УГО – условное графическое обозначение

Введение

Из требований, которые предъявляются к микроконтроллерам приборов можно выделить следующие критерии:

- низкая стоимость;
- высокая надежность;
- малое энергопотребление;
- достаточная производительность для выполнения всех требуемых функций.

Микроконтроллер – это электронное устройство, сочетающее на одном кристалле свойства процессора и периферийные устройства. По своей сути является маломощным компьютером способным выполнять простые задачи. Отличается от микропроцессора интеграцией всей необходимой периферии (таймеров, устройств ввода-вывода, памяти) в один корпус.

Данная выпускная квалификационная работа посвящена разработке универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*. Разработанное устройство предназначено для чтения, копирования, хранения и воспроизведения *RFID*-меток. Управление режимами работы осуществляется за счет трех кнопок. Вывод информации производится на *OLED*-дисплей. В качестве микроконтроллера используется *STM32F030CCT6*.

1 Назначение, состав, область применения и принципы функционирования универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*

Целью данной выпускной квалификационной работы является разработка универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM*32.

Из-за большого распространения домофонных систем и большого разнообразия комбинаций ключей к ним, работникам медицинских, правоохранительных, жилищно-коммунальных служб приходится обращаться за помощью к жильцам, что не всегда возможно. Поэтому наличие ключа облегчит и ускорит работу сотрудников государственных учреждений.

Преимущества технологии *RFID* перед схожими технологиями представлены в сравнительной таблице 1.

Таблица 1 - Сравнение со схожими технологиями

Характеристики	*RFID*	Штрих-код	*QR*-код
Возможность перезаписи	Да	Нет	Нет
Дальность взаимодействия	До 100 метров	До 4 метров	До 1 метра
Длительность жизни	Более 10 лет	Зависит от материала и способа печати	Зависит от материала и способа печати
Объем памяти	10 – 510000 байт	До 100 байт	До 3072 байт
Прямой зрительный контакт	Не обязателен	Необходим	Необходим
Идентификация движущихся объектов	Да	Проблематично	Проблематично

Характеристики	*RFID*	Штрих-код	*QR*-код
Взаимодействие с телом	Да	Проблематично	Проблематично
Идентификация металлических объектов	Да	Да	Да
Работа при повреждении метки	Невозможна	Затруднена	Затруднена
Габаритные характеристики	Средние и малые	Малые	Малые
Стоимость	Средняя и высокая	Низкая	Низкая

Технология *RFID* позволяет читать транспондеры, находящиеся вне прямой видимости от считывателя, например, сейчас стала популярна технология подкожного внедрения *RFID*-меток для быстрой и удобной верификации. Такие метки за частую работают на 13.56 кГц и нужны для передачи данных о носителе (хронические заболевания, адрес проживания и т.д.) или для оплаты покупок.

Таким образом, устройство, которое способно хранить, воспроизводить несколько ключей и читать, обладающее малыми габаритами и автономностью, в данный момент времени актуально.

В соответствии с техническим заданием на проектирование, требуется разработать универсальный ключ идентификации и контроля *RFID*-меток, обладающий следующими параметрами и характеристиками:

- малые габариты;
- автономность;
- хранение большого количества ключей;
- низкая себестоимость.

Для реализации, устройство, которое способно работать автономно, хранить в себе большое количество ключей и будет иметь малый размер были выбраны следующие составляющие:

- *STM32F030CCT6* в качестве ядра устройства;

- *OLED*-дисплей, в качестве устройства вывода интерфейса и информации;

- кнопки в качестве устройства ввода;

- антенна с частотой 125кГц;

- устройство *NE532D* в качестве усилителя для антенны *RFID*;

- *micro-USB* в качестве разъема питания;

- разъем *PLS*-4 для программирования;

- разъем *PLS*-2 для передачи информации;

- разъем *PLS*-4 для подключения *OLED* дисплея;

- устройство *LM3940IMPX* для преобразования 5 В в 3.3 В.

1.1 Обзор используемого оборудования

1.1.1 Микроконтроллер S*TM32F030*

STM32 – микроконтроллер который относится к семейству 32-битных микроконтроллеров производства *STMicroelectronics*. Основан на базе ядра *ARM cortex* различных серий (имена сериям даются от используемого в них ядра *ARM*) и имеет сокращённые наборы инструкций.

Серия *STM32F0* основана на базе ядра *Cortex-M0* является микроконтроллером широкого применения и обладает следующими характеристиками:

- характеристики ядра:

  1) 32-разрядный процессор *Arm® Cortex®-M0*;

  2) частота до 48 МГц;

- характеристики памяти:

  1) от 16 до 256 Кбайт флэш-памяти;

  2) от 4 до 32 Кбайт *SRAM* с аппаратной четностью;

- блок вычисления *CRC*;

- сброс и управление питанием;

- напряжение цифровых линий (входа/выхода): зависит от напряжения питания, от 2,4 В до 3,6 В;

- аналоговое питание: *VDDA = VDD* до 3,6 В;

- сброс при включении/отключении питания (*POR/PDR*);

- режимы низкого энергопотребления: сон, стоп, режим ожидания;

- управление часами;

- кварцевый генератор от 4 до 32 МГц;

- генератор 32 кГц для *RTC* с калибровкой;

- внутренний *RC* 8 МГц с опцией x6 *PLL*;

- внутренний *RC*-генератор 40 кГц;

- пятиканальный контроллер прямого доступа к памяти;

- один двенадцатиразрядный АЦП с временем отклика 1,0 мкс (до 16 каналов);

- календарь *RTC* с будильником и периодическим пробуждением;

- один шестнадцати-битный таймер с расширенным управлением;

- шестиканальный ШИМ-выход;

- *IC/OC*, *OCN*, можно использовать для ИК-управления;

- независимые и системные сторожевые таймеры;

- таймер *SysTick* [2].

Есть несколько основных факторов из-за чего продукты на базе *STM32F0* имеют более низкую стоимость по сравнению с аналогичными на восьми битных и шестнадцати битных контроллерах. Более современной технологии производства. Благодаря оптимизированному алгоритму выполнения команд, выполнение операций занимает меньше времени. Чем меньше места требуется для программы, тем за меньшие количества тактов выполняется программа, и тем легче производить их отладку. А это существенно сокращает время, необходимое для проведения разработки. Все эти преимущества приводят к снижению стоимости продуктов, основанных на базе *STM32*.

Микроконтроллеры серии *STM32F0* мало чем отличаются от микроконтроллеров серии *STM32F1* на основе ядра *Cortex M3*. Основным недостатком серии *F0* является усеченный набор команд. Это означает что программное обеспечение,

написанное для *STM32F0*, будет в полном объеме работать и на контроллерах *STM32F1* и более старших семействах микроконтроллеров, благодаря тому, что ядра *Cortex-M3* и *Cortex-M4* включают все инструкции *Cortex-M0*. И, в случае нехватки производительности у контроллеров *STM32F0*, в любой момент можно перенести разработку на более мощный микроконтроллер. Но в тоже время с более старших моделей нельзя будет перенести программу на серию *STM32F0.*

Микроконтроллеры *STM32F030x4/6/8/C* содержат высокопроизводительный процессор *ARM®. Cortex®-M0* 32-битное *RISC*-ядро, работающее на частоте 48 МГц, встроенную память (до 256 Кбайт флэш-памяти и до 32 Кбайт *SRAM*) и широкий спектр периферийных устройств. Все устройства предлагают стандартные коммуникационные интерфейсы (до двух *I2C*, до двух *SPI* и до шести *USART*), один 12-битный АЦП, семь 16-разрядных таймеров общего назначения и ШИМ-таймер с расширенным управлением. Микроконтроллеры *STM32F030x4/6/8/C* работают в диапазоне температур от -40 до +85 °C. от источника питания 2,4-3,6 В. Полный набор режимов энергосбережения позволяет проектирование маломощных приложений. Микроконтроллеры *STM32F030x4/6/8/C* включают устройства в четырех различных корпусах. от 20 до 64 контактов. Разные модели и конфигурации микроконтроллеров имеют разное количество периферийных устройств. Благодаря низкому энергопотреблению и компактности микроконтроллеры серии *STM32F030x4/6/8/C* отлично подходят для применения их в портативных *GPS* устройствах, устройствах регистрации, устройствах контроля. Модель *STM32F030* с 48 контактами представлен на рисунке 1.1.

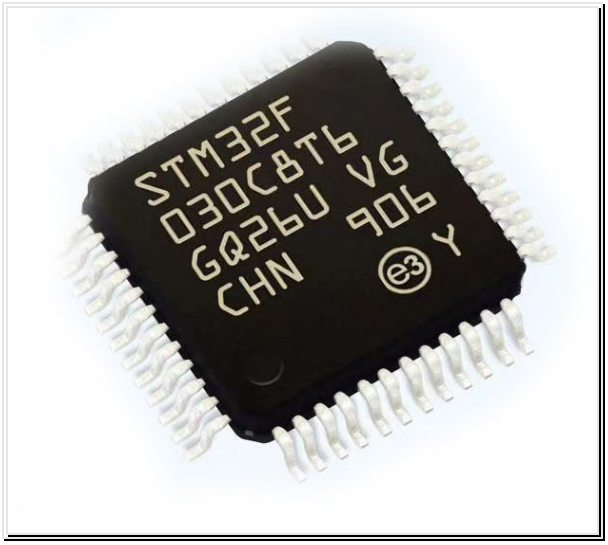


Рисунок 1.1 – Микроконтроллер *STM32F030C8T6*

1.1.2 Операционный усилитель *NE532D*

NE532D состоит из двух независимых операционных усилителей с высоким коэффициентом усиления и внутренней частотной компенсацией. Операционные усилители с внутренней частотной компенсацией разработаны специально для работы от одного источника питания в широком диапазоне напряжений.

Входы усилителя имеют высокое сопротивление. На входах происходит оценка величины напряжения и в зависимости от оценки величин усилитель выдает на выход измененный сигнал.

Существуют несколько разных схем операционного усилителя:

- инвертирующий, характеризуется изменением фазы входного сигнала на противоположную;

- не инвертирующий, усиливает входящий сигнал не изменяя фазу;

- дифференциатор, принцип работы такой схемы зависит от разницы скорости изменения сигнала, чем больше скорость изменения сигнала, тем больше величина коэффициента усиления;

- аналоговый компаратор с гистерезисом, представляет собой аналоговый компаратор, к выходу которого подключен вход делителя напряжения, а к минусовому контакту подключен выход делителя напряжения.

Рисунок 1.2 –Операционный усилитель *NE532D*

1.1.3 Стабилизатор напряжения *LM3940IMPX*

Линейный преобразователь работает как регулируемый делитель напряжения. На вход делителя подаётся входное напряжение, а выходное снимается с одного из плеч делителя.

Одно из плеч постоянно корректирует сопротивление и тем самым гасит лишнее напряжение.

LM3940 — это преобразователь с малым падением напряжения (0.2 В), разработанный для обеспечения 3,3 В от источника питания 5 В. *LM3940* идеально подходит для систем, питающихся от 5 и от 3 В одновременно.

Корпус TO-220 *LM3940* означает, что в большинстве компоновок можно использовать полный ток нагрузки 1А. Поставляется без использования дополнительного радиатора. В корпусе *DDPAK/TO-263* для поверхностного монтажа используется минимальное пространство на плате и дает отличную мощность рассеивания при пайке к медной площадке на печатной плате. Внутренние устройства *LM3940IMPX* представлено на рисунке 1.3.

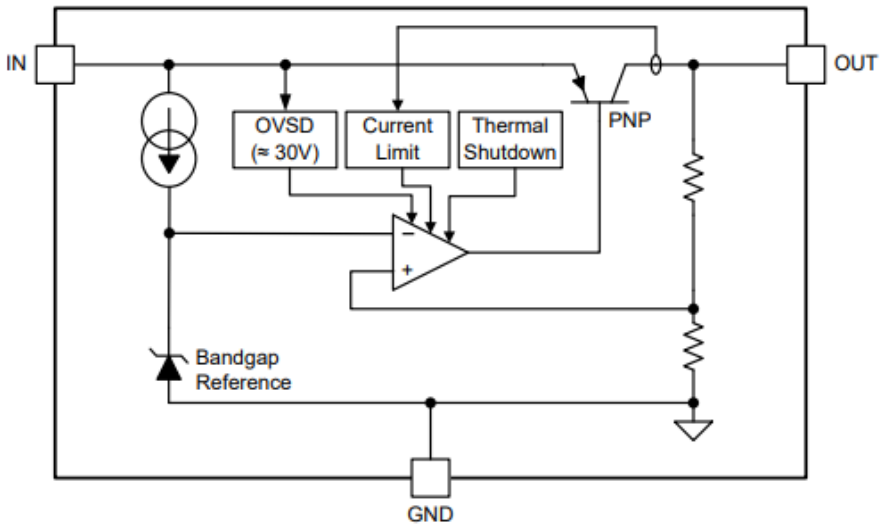


Рисунок 1.3 –Стабилизатор напряжения *LM3940IMPX*

1.2 Принципы функционирования RFID

RFID (Radio Frequency IDentification) – радиочастотная идентификация. *RFID* технологии применяются для идентификации физических объектов и отслеживания их перемещений. На радиочастотную метку можно записать информацию чтобы в последующим ее считать.

RFID – метка, это устройство на которое записывается какая либо информация. В последствие информацию может считать *RFID* – считыватель. Транспондер иметь различные формы и размеры.

Метка состоит из двух частей:

- антенна - принимает радиочастотный сигнал;

- микрочип - отвечает за хранение данных.

RFID-метки разделяются по принципам работы:

- пассивные, у них отсутствует источник питания. Поэтому они работают только при контакте со считывателем;

- полупассивные, имеют свой источник питания. Благодаря этому они работают без воздействия считывателя и на более большом расстоянии, но выходят из строя после разряда источника питания;

- активные также имеют источник питания, но на много мощнее. Дистанция

взаимодействия со считывателем достигает 50 метров.

Разделение по используемой памяти:

- *RO (read only)* – содержат блок идентификации. Никакую новую информацию на них записать нельзя;

- *WORM (Write Once Read Many)* – помимо идентификатора имеет блок памяти для однократной записи. Эту информацию можно многократно считывать;

- *RW (read and write)* – эти метки можно перезаписывать и многократно читать.

Различия по рабочей частоте:

- *LF* с диапазоном 125 – 134 кГц – одни из самых распространённых систем, из-за длина волны работают на малом расстоянии от считывателя;

- *HF* с диапазоном 13,56 МГц – применяются в платежных системах, системах логистики, системах идентификации личности. Для данной частоты разработан стандарт *ISO 14443*. Этот стандарт обеспечивает разнообразие ключей, что позволяет создавать открытые системы. Так же, как и в LF системах существуют проблемы с расстоянием взаимодействия;

- *UHF* с диапазоном 860 – 960 МГц – такие метки обладают наибольшей дальностью регистрации. Разрабатывались для логистики на складах. С 2008 года стали соответствовать следующим критериям: обеспечение уникальности метки, возможность производителю товара записывать произвольные *EPC*-номера. В отличие от систем, основанных на *LF* и *HF* метки в *UHF*, стоят дешевле, но стоимость прочего оборудования выше.

Взаимодействие *RFID* устройств основано на прицепе индукционных связей и принципе обратного рассеивания.

Принцип индукционной связи заключается в том, что небольшая часть излучаемого поля пронизывает витки антенны транспондера, находящегося на некотором удалении от антенны считывателя. За счет электромагнитной индукции на катушке-приемнике считывателя возникает напряжение, которое выпрямляется и применяется в качестве питания микрочипа. Параллельно обмотке антенны под-

ключается емкость, которая в цепи с антенной формирует параллельную резонирующую цепь, частота резонанса которой соответствует частоте передаваемого сигнала.

Индуктивно связанные системы основаны на связи трансформаторного типа между первичной обмоткой считывателя и вторичной обмоткой метки. В случае, когда резонирующая метка находится в пределах переменного магнитного поля антенны считывателя, она забирает часть энергии магнитного поля. Принцип работы индуктивной связи в устройствах *RFID* представлен на рисунке 1.4.

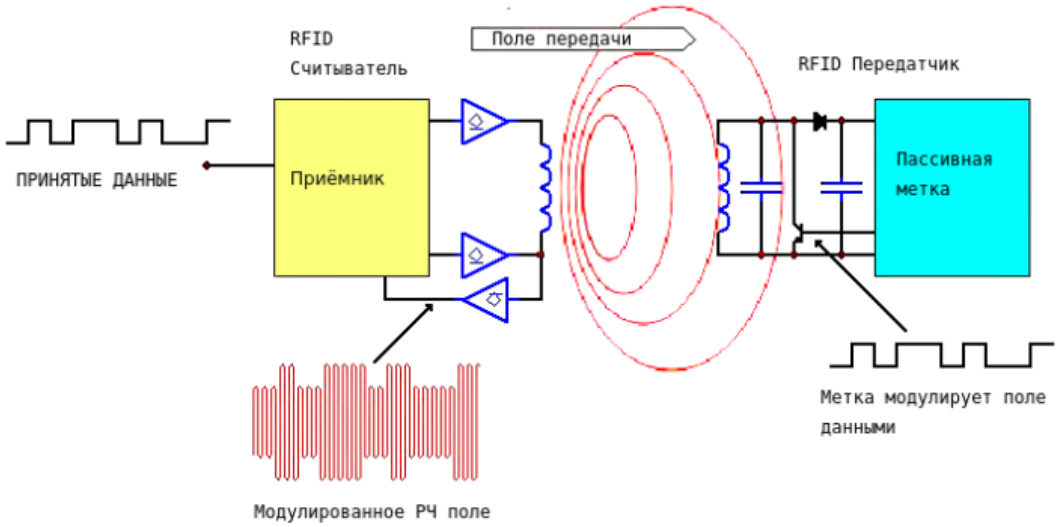


Рисунок 1.4 – Принцип работы *RFID*

Для восстановления данных на стороне считывателя, напряжение, измеренное на его антенне, выпрямляется. Таким образом производится демодуляция амплитуды модулированного сигнала.

1.3 Принципы функционирования универсального ключа идентификации и контроля RFID-меток на основе микроконтроллера STM32

Разрабатываемое устройство, универсальный ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*, предназначено для чтения, копирования, хранения и воспроизведения *RFID* меток. Чтение обеспечивает ка-

тушка индуктивности, которая улавливает электромагнитные волны на определенной частоте (125 – 130 МГц). Также при помощи этой же катушки и двух мосфетов (*Mosfet*) генерируется 16 битный *RFID* ключ. После успешной генерации или же копирования ключа, прибор предлагает его сохранить в память.

Управление устройством осуществляется за счет нажатия на 3 кнопки. Левая и средняя перемещают курсор. Нажатие на правую совместно с кнопками управления позволяет (правая средняя) перейти в выбранный пункт, (левая правая) вернуться назад.

Устройство питается от +5 В. Для преобразования напряжения в необходимые для работы микроконтроллеров +3.3 В используется стабилизатор напряжения *LM3940IMPX*.

1.4 Разработка схемы электрической структурной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*

Схема электрическая структурная универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32* приведена на рисунке 1.5 (см. также приложение А, первый лист графической части выпускной квалификационной работы).

Схема содержит следующие основные блоки:

- БС – блок синхронизации;
- БСН – блок стабилизации напряжения;
- БУ – блок управления;
- БМК – блок микроконтроллера;
- БД – блок дисплея;
- БГЧК– блок генерации/чтения ключа *RFID;*
- БИ – блок индикации;
- БА– блок антенны.

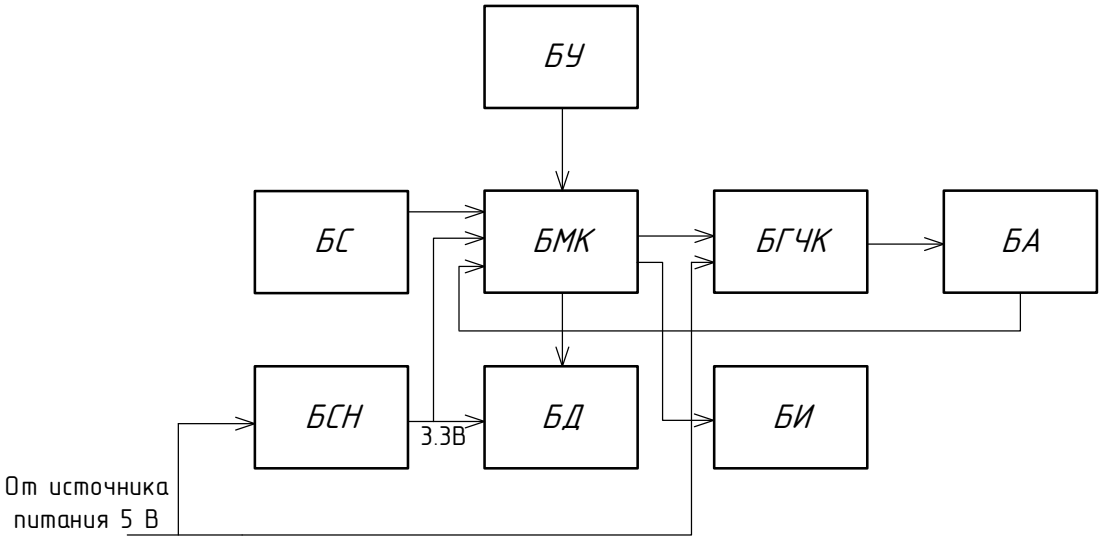


Рисунок 1.5 – Схема электрическая структурная

Блок стабилизации напряжения формирует стабилизированный уровень питающего напряжения, необходимый для обеспечения работоспособности БМК и БД. Напряжение питания на входе блока стабилизатора (+5 В), напряжение поступает с разъема micro-*USB*.

Блок микроконтроллера содержит непосредственно микроконтроллер и является ядром всей микроконтроллерной системы и выполняет все основные функции по управлению работой универсального ключа идентификации и контроля *RFID*-меток. (Управляет выводом на дисплей, генерацией ключа и приемом ключа на катушку *RFID*).

Блок дисплея служит для отображения информации о режимах работы и хранящихся в памяти *RFID* меток.

При включении питания происходит инициализация микроконтроллерного устройства (чтение адреса первой исполняемой команды программы из внутренней памяти программ, конфигурирование портов ввода-вывода, инициализация регистров и т.п.). После чего микроконтроллер выводит меню на дисплей и ожидает выбора режима работы.

Блок синхронизации предназначен для формирования последовательности синхронизирующих импульсов, необходимых для тактирования работы микроконтроллера.

Блок генерации/чтения ключа *RFID* генерирует и выводит на нужную частоту сигнал ключа, а также принимает возвратный сигнал от транспондера.

Блок индикации – служит для отладки и вывода кодов ошибки при повреждении или невозможности использования *OLED* дисплея.

1.5 Выбор среды разработки

На рынке программного обеспечения присутствует четыре самых популярных продукта для разработки моделирования, и проектирования печатных плат:

- *Altium Designer;*
- *P-CAD;*
- *PADS Professional;*
- *Sprint-Layout.*

Altium Designer – отличный набор инструментов для разработки и проектирования принципиальных схем и печатных плат. Обладает удобным интерфейсом и высокой производительностью. Благодаря поддержке *DirectX* произошло перераспределение нагрузки между процессором и видеокартой. Имеет инструментарий для проектирования библиотек УГО и посадочных мест, из которых в последствии удобно создавать принципиальные схемы. Принципиальные схемы могут быть преобразованы в *PCB*, все связи между компонентами установленные в принципиальной схеме сохраняются и подсвечиваются при расположении компонентов на печатной плате.

P-CAD – разработан той же компанией что и *Altium Designer*, но его поддержка была прекращена в 2008 году. Не обладает таким удобным интерфейсом как *Altium*.

Sprint-Layout – простая среда, предназначенная для проектирования печатных плат при помощи библиотек уже загруженных в среду, обладает достаточно удобным и простым интерфейсом, но из-за малого инструментария не подходит для профессиональной работы.

PADS Professional от *Mentor Graphics*, ее основным и преимуществом перед

другими программами, является возможность децентрализованной работы нескольких инженеров над одним проектом, это достигается с помощью клиент-серверной системы. Также большим преимуществом является возможность параллельной работы над проектом, то есть несколько инженеров одновременно могут заниматься разработкой, проектированием и логистикой проекта. В случае ошибки предусмотрена система автоматического резервного копирования проекта (*Project backup*), эта система позволяет быстро откатить проект и продолжить работу с момента последнего сохранения. Исходя из перечисленного можно сравнить *PADS Professional* с не менее популярной средой *Altium Designer*. Благодаря приспособленности к децентрализованной работе и клиент-серверной архитектуре среду *PADS* можно считать лучшей средой для больших компаний с широким штатом специалистов.

Исходя из выше перечисленного и руководствуясь сравнительной таблицей 2 можно прийти к выводу, что среда *PADS* обладает большим функционалом перед своими аналогами. Даже несмотря на менее простой интерфейс в сравнение с *Altium Designer*, возможность удаленного проектирования, а также возможность работы большого количества специалистов над одним проектом делает эту среду более предпочтительной перед конкурентами.

Таблица 2 - Сравнение средств разработки и проектирования

	Разработка схем	Проектирование плат	Создание библиотеке	Возможность работать удаленно
Altium Designer	+	+	+	-
P-CAD	+	+	+	-
PADS Professional	+	+	+	+
Sprint-Layout	-	+	-	-

1.6 Разработка схемы электрической принципиальной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*

1.6.1 Выбор элементной базы

Выбор между микроконтроллерами различных фирм часто бывает затруднителен. Нередко похожие по характеристикам микроконтроллеры можно встретить в каталогах разных фирм. При этом, как правило, фирмы обеспечивают эффективную поддержку разработчиков, включающую предоставление технических материалов и бесплатного программного обеспечения для разработки программ.

Для реализации универсального ключа идентификации и контроля *RFID*-меток был выбран микроконтроллер *STM32F030*, основанный на архитектуре *ARM*.

Перед *ARM* стояла цель создать архитектуру, лишенную недостатков x86. В результате у них получился крайне эффективный и производительный продукт. Популярности *ARM*-процессоров на рынке мобильных устройств способствовало не только это, но также ряд других преимуществ:

- микроконтроллеры *STM32* дешевле в производстве и развертывании;
- архитектура *ARM* позволяет крупным компаниям создать свои собственные решения на *ARM*-архитектуре для разных ниш;
- гибкие возможности настройки.

Кроме того, эффективное потребление энергии и отсутствие перегрева делают такие процессоры оптимальным выбором для серверных задач, а также для использования в маршрутизаторах и высокопроизводительных решениях хранения данных.

В свою очередь *STM32F030C8T6* обладает следующими характеристиками:

- характеристики ядра:
    1) 2-разрядный процессор *Arm® Cortex®-M0*;
    2) частота до 48 МГц;
- характеристики памяти:

1) от 16 до 256 Кбайт флэш-памяти;

2) от 4 до 32 Кбайт *SRAM* с аппаратной четностью;

- блок вычисления CRC;

- сброс и управление питанием;

- напряжение цифровых линий (входа/выхода): зависит от напряжения питания, от 2,4 В до 3,6 В;

- аналоговое питание: *VDDA = VDD* до 3,6 В;

- сброс при включении/отключении питания (*POR/PDR*);

- режимы низкого энергопотребления: сон, стоп, режим ожидания;

- управление часами;

- кварцевый генератор от 4 до 32 МГц;

- генератор 32 кГц для *RTC* с калибровкой;

- внутренний *RC* 8 МГц с опцией x6 *PLL*;

- внутренний *RC*-генератор 40 кГц;

- пятиканальный контроллер прямого доступа к памяти;

- один двенадцатиразрядный АЦП с временем отклика 1,0 мкс (до 16 каналов);

- отдельное аналоговый питание: от 2,4 В до 3,6 В;

- календарь *RTC* с будильником и периодическим пробуждением;

- один шестнадцатибитный таймер с расширенным управлением;

- шестиканальный ШИМ-выход;

- *IC/OC*, *OCN*, можно использовать для ИК-управления;

- независимые и системные сторожевые таймеры;

- таймер *SysTick* [2].

Микроконтроллер имеет 2 внутренних счетчика которые синхронизируются с внешними кварцевыми резонаторами 8 МГц и 32.768 кГц. Программирование устройства осуществляется через специальный разъем *pls*-4 который подключен к *SWD* микроконтроллера, для передачи информации от устройства используется разъем *pls*-2 подключенный *UART*, для взаимодействия с *OLED*-дисплеем исполь-

зуется *pls*-4, подключенный к интерфейсу *I2C*.

Стоит учитывать, что микроконтроллер *STM32* имеет три режима загрузки:

- основная флеш-память;

- системная память;

- встроенная *SRAM*.

Режимы загрузки задаются путем выставления уровня на порты, имеющие функционал *BOOT0* и *BOOT1*. Для работы устройства нас интересует нормальный режим.

Для приведения сигнала к логическому виду на входе и выходе сигнальных контактов стоят триггеры Шмитта.

Для генерации ключа были выбраны два мосфета *(Mosfet*): один n-канальный, второй p-канальный; на затвор мосфетов *(Mosfet)* подается сигнал с микроконтроллера. Благодаря открытию мосфетов *(Mosfet)* происходит генерация логического сигнала, который является ключом, где высокий уровень — это единица, а низкий-ноль.

Для управления режимами работы устройства на плате установлены три кнопки, которые при нажатии подают высокий уровень на входы микроконтроллера. Разные комбинации клавиш задают разные режимы работы.

Для отладки, проверки и отображения работы сигналов используется БИ. На светодиоды выводится копия последовательности сигнала, идущего на мосфеты *(Mosfet)*. В случае ошибки на светодиоды поступает код ошибки.

На рисунке 1.6 представлены режимы загрузки, а на рисунке 1.7 представлена функциональная схема микроконтроллера *STM*32.

mode	select		decription
	boot 0	boot 1	
Основная флеш-память	0	X	Нормальный режим, неизбежный выбор для запуска пользовательских программ
Системная память	1	0	Запустите загрузчик, макроскопически говоря, эта память предоставляет программу загрузки USB. Мы можем запустить эту программу загрузки, чтобы загрузить программу во FLASH через USB. Обратите внимание, что она предназначена для загрузки во FLASH. Если нам нужно запустить только что загруженную программу, мы должны преобразовать Перейдите в нормальный режим (начните с FLASH), но вы можете разработать схему для реализации загрузки одним нажатием. Смотрите продолжение ...
Встроенная SRAM	1	1	Этот режим используется для отладки и не сохраняется после выключения питания; уменьшите количество стирания FLASH

Рисунок 1.6 – Режимы загрузки *STM32*

OLED дисплей подключается к шине *I2C. I2C* - синхронная, состоит из двух линий: данных (*SDA*) и тактирования (*SCL*). Есть ведущий и ведомые.

Такты на линии *SCL* генерирует ведущий. Линией *SDA* могут управлять как ведущий, так и ведомый в зависимости от направления передачи. Обмен происходит при помощи пакетов, обрамленными уникальными условиями на шине, именуемыми стартовым и стоповым условиями. Ведущий в начале каждого пакета передает один байт, где указывает адрес ведомого и направление передачи последующих данных. Данные передаются 8-битными словами. После каждого слова передается один бит подтверждения приема приемной стороной.

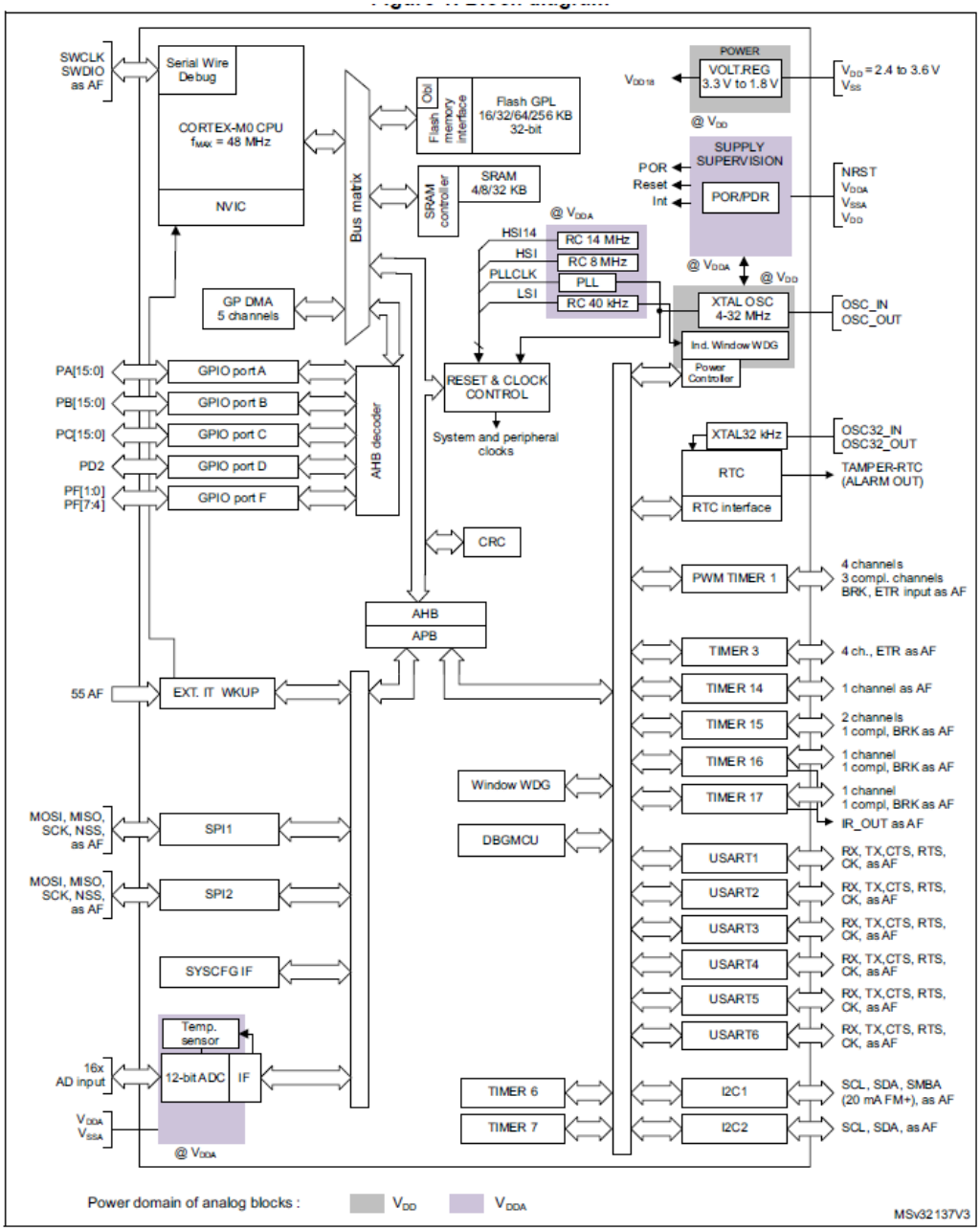


Рисунок 1.7 – Функциональная схема микроконтроллера *STM32F030C8T6*

Условное графическое изображение микросхемы микроконтроллера *STM32F030C8T6* приведено на рисунке 1.8.

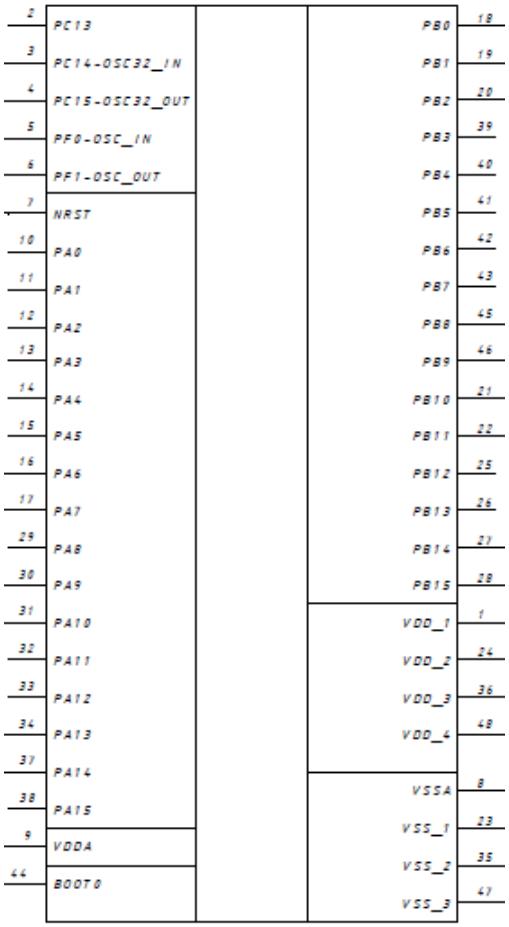


Рисунок 1.8 – УГО микроконтроллера *STM32F030C8T6*

Условное графическое изображение микросхемы *LM*3940 и типовая схема включения приведены на рисунке 1.9.

Микросхема *LM3940* – 4 выводной стабилизатор с фиксированным выходным напряжением 3.3 В и током в 1 А.

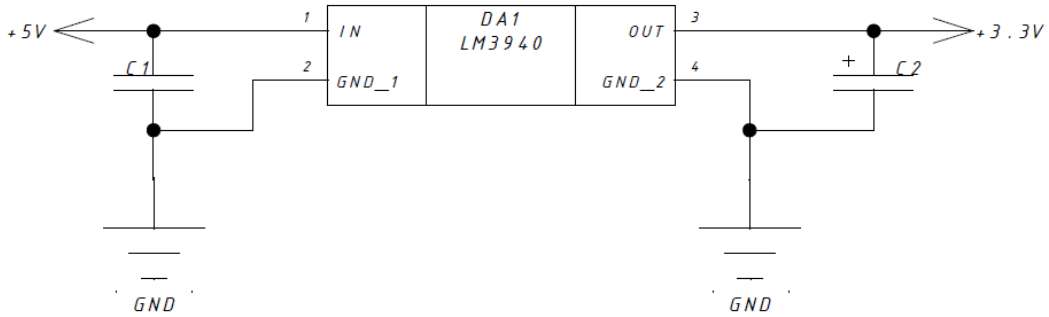


Рисунок 1.9 – Условное графическое изображение микросхемы *LM3940*

Основные характеристики *LM3940*:

- диапазон входного напряжения: от 4,5 до 5,5 В;
- выходное напряжение в зависимости от температуры;
- превосходное регулирование нагрузки;
- указанный выходной ток 1 А;
- требуется только один внешний компонент;
- встроенная защита от перегрева;
- защита от короткого замыкания;
- Условное графическое изображение микросхемы *SN74LVC1G17DBVT* и типовая схема включения представлены на рисунке 1.10 и на рисунке 1.11 соответственно.

Микросхема *SN74LVC1G17DBVT* – триггер Шмитта, помещенный в корпус *SOT-23*.

Основные характеристики *SN74LVC1G17DBVT*:

- диапазон входного напряжения 1.65 – 5.5 В;
- максимальный выходной ток высокого уровня 32 мА;
- максимальное время задержки при максимальном *CL 5 ns* при 5 В.

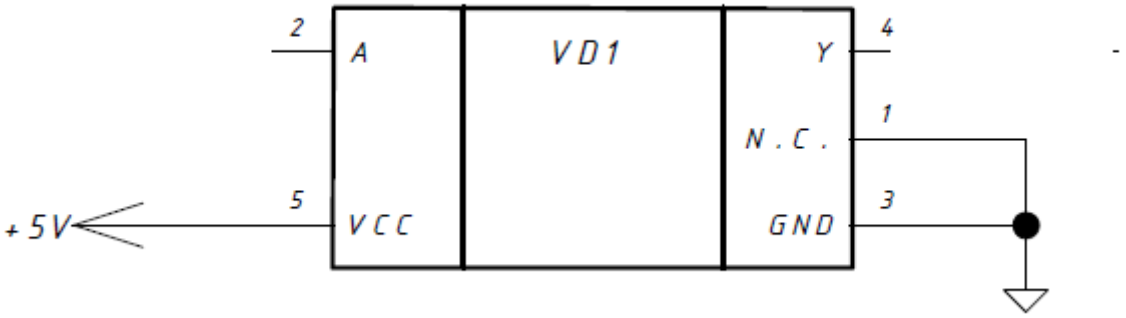


Рисунок 1.10 – Условное графическое изображение триггера Шмитта *SN74LVC1G17DBVT* на выходе микроконтроллера

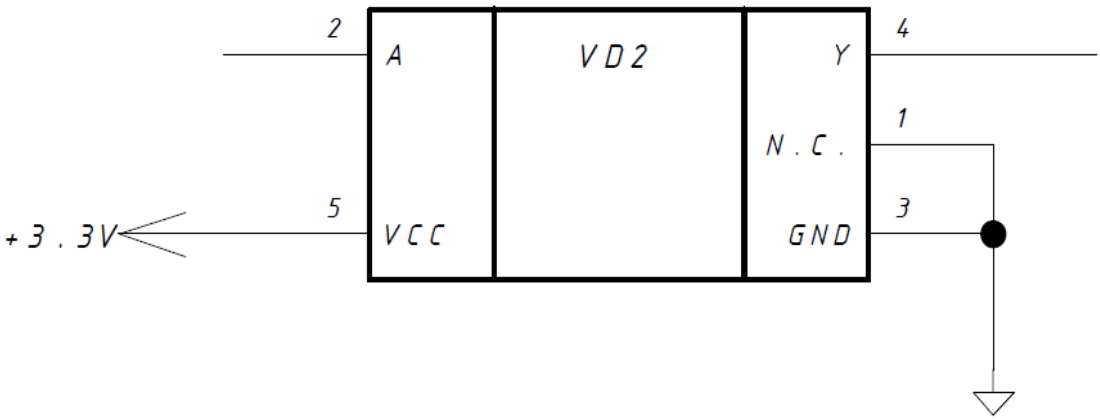


Рисунок 1.11 – Условно графическое изображение триггера Шмитта *SN74LVC1G17DBVT* на входе микроконтроллера

Следует обратить внимание что питание у продемонстрированных триггеров Шмитта различно при подключении к входу и выходу микроконтроллера.

Диод Шоттки - это полупроводниковый диод, который характеризуется минимальным падением напряжения на прямо смещенным переходе. Диод Шоттки создается на основе перехода металл-полупроводник, поэтому отличается высоким быстродействием. В зависимости от того, какое конкретно вещество выбрано для изготовления перехода в элементе, различаются и характеристики элемента. Чаще всего выбирается кремний, возможно применение арсенида галлия. Реже могут применяться сплавы вольфрама, платины и других материалов.

Условное графическое изображение микросхемы *NE532D* изображена на рисунке 1.12

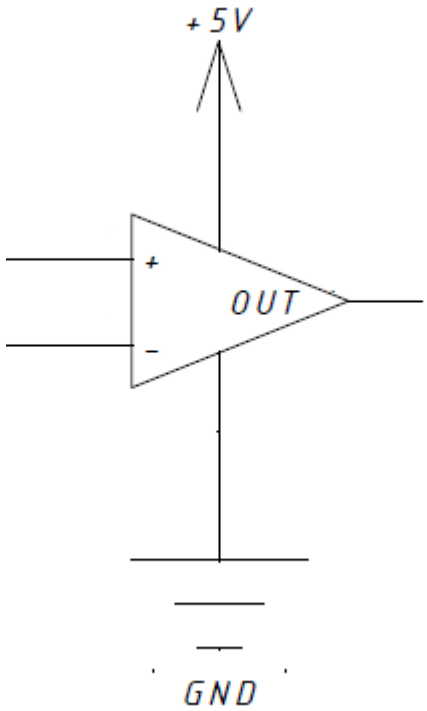


Рисунок 1.12 – Условное графическое изображение микросхемы NE532D

Микросхема *NE532D* – восьмивыводной операционный усилитель (два усилителя в одном корпусе) с общим питанием. Для удобства в проектировании схемы был разделен на части.

Основные характеристики *NE532D*:

- внутренняя частотная компенсация для единичного усиления;
- большое усиление постоянного напряжения: 100 дБ;
- широкая полоса пропускания (единичное усиление): 1 МГц (с температурной компенсацией);
- широкий диапазон однополярного питания: от 3 до 30 В;
- использования двух полярного источника питания: от ±1,5 В до ±15 В;
- низкий потребляемый ток питания (400 мкА);
- низкий входной ток смещения: 45 нА постоянного тока с температурной компенсацией;

- диапазон дифференциального входного напряжения равен напряжению источника питания.

1.6.2 Создание библиотеки УГО

Перед началом работы была разработана библиотека элементов при помощи инструмента *Xpedition Library Manager*. Отображение УГО *STM32* в среде *PADS* представлено на рисунке 1.13.

В первую очередь была разработана библиотека основных элементов УГО в соответствии с ГОСТом.

Затем были созданы шаблоны контактных площадок при помощи инструмента *Padstack Editor*.

После создания библиотек УГО были созданы библиотеки посадочных мест. Рассмотрим пример библиотеки все на том же микроконтроллере.

Посадочное место состоит из нескольких слоев обозначенных разными цветами:

- *Silkscreen Outline* – слой шелкографии имеет толщину в 0,2 мм, срезанный угол и точка в левом нижнем углу обозначает точку отсчета контактов, контакты отсчитываются против часовой от точки, скоса или черты;

- *Assembly Outline* – слой для обозначения на сборочном чертеже имеет толщину в 0,1 мм, в остальном полностью совпадает со слоем шелкографии;

- *Placement Outline* – слой необходимый для отображения контуров элемента для последующего аппаратного расположения их на плате.

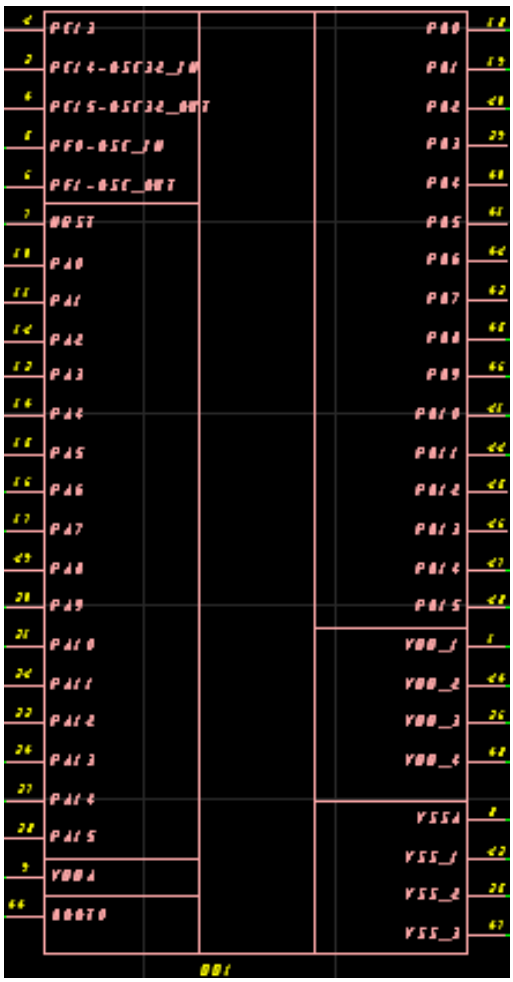


Рисунок 1.13 – УГО микроконтроллера *STM32* в среде *PADS*

Слой металлизации под контакты микроконтроллера имеют размер на 0,1 мм больше для более удобного монтажа, так же для этого по контуру площадки был снят 0,1 мм защитной маски.

На рисунке 1.14 будет изображено меню настройки посадочного мета для *SMD* компонентов.

На рисунке 1.15 будет изображено готовое посадочное место для микроконтроллера *STM32*.

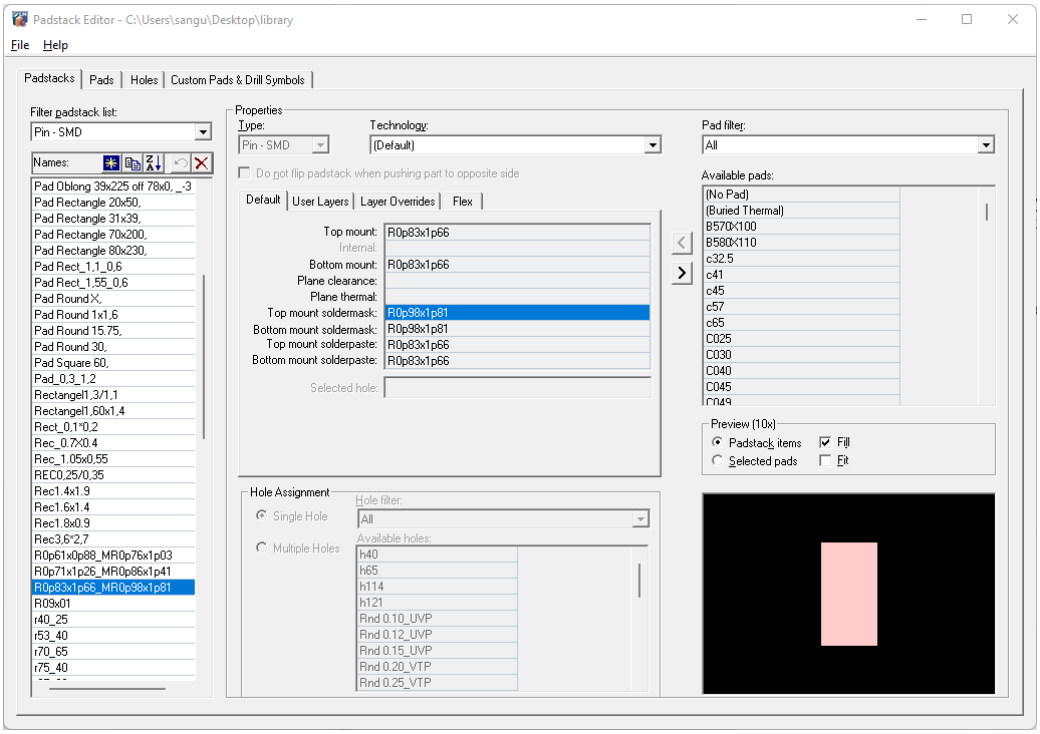


Рисунок 1.14 – Инструмент *Padstack Editor*

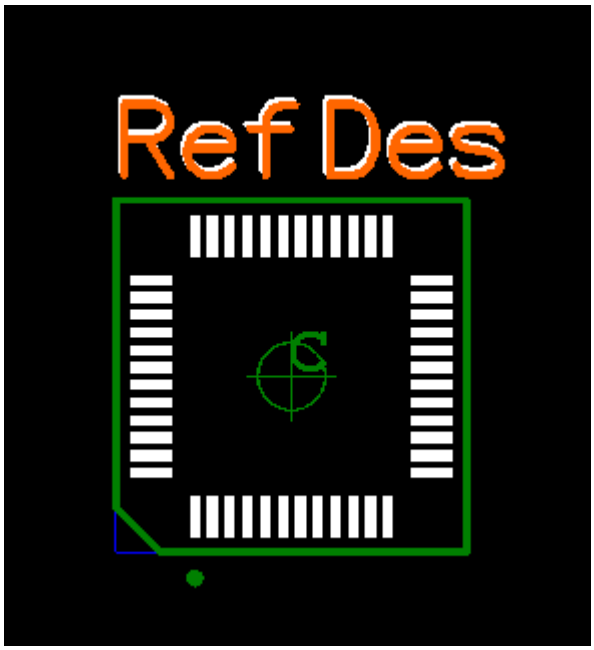


Рисунок 1.15 – Посадочное место микроконтроллера *STM32*

1.6.3 Описание схемы электрической принципиальной универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM*32

Схема электрическая принципиальная универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32* приведена в приложении В (второй лист графической части выпускной квалификационной работы). Схема электрическая принципиальная содержит следующие микросхемы и электрорадиоэлементы:

- линейный преобразователь *LM3940IMPX (DA1*);
- операционный усилитель *NE532D (DA2*);
- микроконтроллер *STM32F030CCT6 (DD1*);
- триггер Шмитта *SN74LVC1G17DBVT (VD1, VD2*);
- кнопки *MJTP1141 (SW1…SW4);*
- разъемы *PLS-2 (X1, X5*), *PLS-4 (X2, X4), ADAMTECH_MCR-B-S-RA-SMT-CS14-T_R (X3)*. Один из разъемов *PLS*-2 используется для передачи информации от устройства к компьютеру второй служит для подключения внешней катушки индуктивности. *PLS*-4 (*X2*) используется для программирования микроконтроллера, *PLS*-4 (*X4*) предназначен для подключения *OLED* дисплея. *ADAMTECH_MCR-B-S-RA-SMT-CS14-T_R* разъем *micro-usb* предназначенный для питания устройства;
- кварцевые резонаторы *QT2M-32.768KHZB-T (Q1), LFXTAL05566 (Q2);*
- мосфеты (*mosfet) BS250F (Q3), BS170FTA (Q4);*
- светодиоды *YR1S2 (L1, L3… L5);*
- дроссель *BLM18PG221SN1D* (*L2*) стоит на аналоговом входе и фильтрует питание;
- диоды Шоттке *BAT43* (*L6, L7*);
- резисторы (*R1… R9*);
- конденсаторы (*C1…C18*).

1.6.4 Расчётная часть

Согласно документации, на преобразователь *LM3940* на входе (+5 В) должен стоять конденсатор (*C1*) номиналом в 4,7 *nF* для фильтрации высокочастотных помех, на выходе для стабилизации входного напряжения (+3.3 В) используется танталовый конденсатор (*C2*) номиналом в 33 *nF* для нормализации просадок по питанию прибора. К питанию в 3,3В подключены:

- микроконтроллер *STM32F030CCT6 (DD1*);
- *OLED* дисплей, подключенный к разъему *pls-4 (X4*);
- триггер Шмитта (*VD2*).

Для синхронизации к микроконтроллеру подключены 2 кварцевых резонатора разной частоты 8 МГц и 32,768 кГц для синхронизации. У каждого резонатора стоит по 2 конденсатора номиналом 10 *pF*.

Также к микроконтроллеру подключены 4 кнопки: (*SW2…SW4*) кнопки для управления ПО и кнопка (*SW1*) для сброса; для предотвращения дребезга контактов у каждой кнопки установлены конденсаторы номиналом 0,1 *uF*. Для контроля сигналов и вывода ошибок используется четыре светодиода (*L1, L3…L5*); к каждому из них по питанию подключен резистор. Для выбора режима загрузки стартовой программы используется контакт *BOOT*; выбор пал на стандартный режим загрузки.

Микроконтроллер напрямую подключен к *OLED* дисплею через разъем *pls-4*.

Для генерации ключа микроконтроллер подключен к двум мосфетам *(Mosfet)* (*Q3, Q4*) через триггер Шмитта питающийся от 5 В, после мосфетов *(Mosfet)* сигнал поступает на разъем *pls*-2 к которому подключается антенна.

Для считывания сигнала используется та же самая антенна, с которой сигнал идет на каскад операционного усилителя, где сигнал усиливается до необходимого уровня, после чего сигнал через триггер Шмитта, питающийся от 3,3В, поступает на микроконтроллер.

1.6.5 Расчет блока управления *RFID*

Расчёт резистора на затворе мосфетов (mosfet) см. Рисунок 1.16:

- напряжения затвора = 5 В;

- ток затвора = 25 мА.

Исходя из закона Ома, можно рассчитать необходимый наминал резистора.

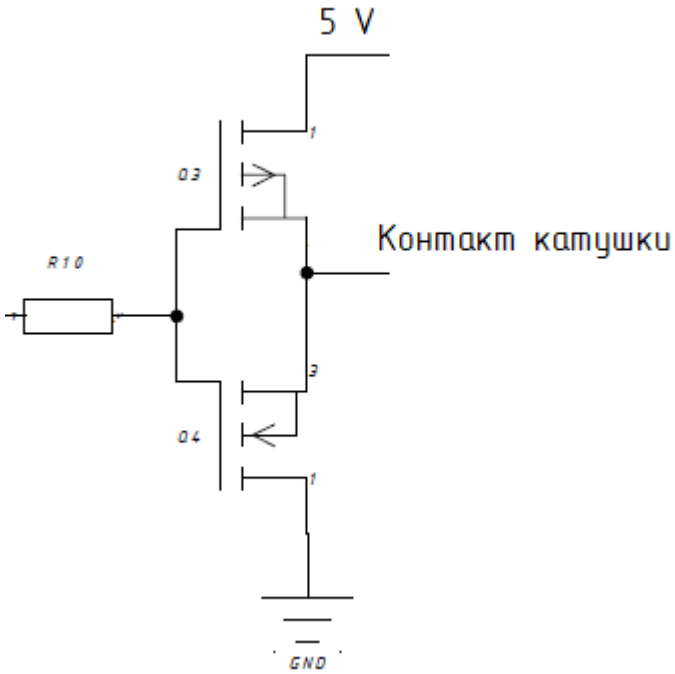


Рисунок 1.16 – Мосфеты (*Mosfet*) генерирующие импульс

$$R10 = (5В/25мА) = 2{,}1 \text{ кОм}.$$

У каждой кнопки стоит подтягивающий резистор, и конденсатор для гашения дребезга:

- напряжения затвора = 5 В;

- ток затвора = 25 мА.

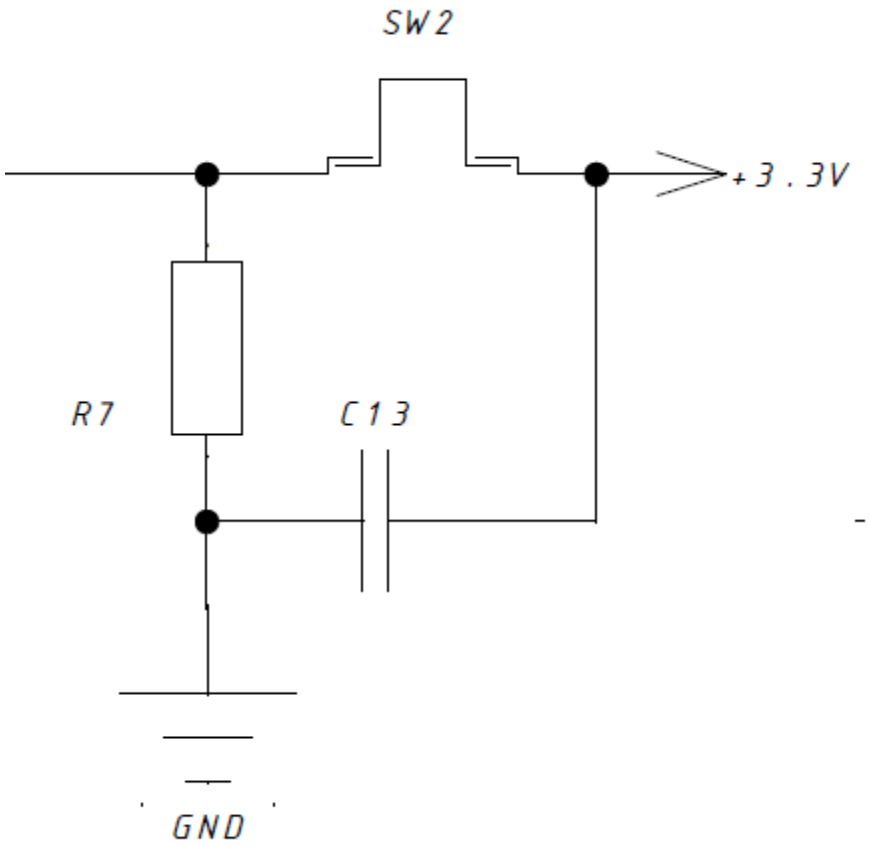


Рисунок 1.17 – Блок управления

Исходя из закона Ома, можно рассчитать необходимый наминал резистора.

R7…R9 = (5В/25мА)*3 = 6,3 кОм.

1.7 Разработка программного обеспечения

1.7.1 Выбор среды разработки ПО

В качестве инструмента разработки ПО был выбран *STM32CubeIDE*, как удобная среда для разработки и программирования микроконтроллера *STM32*. Для программирования устройства достаточно подключить его к компьютеру через разъем *micro-usb*, поле чего загрузить прошивку на микроконтроллер. В случае неисправности линий программирования у разъема, можно подключиться на прямую к линиям *BOOT* через перемычки *S5*, *S6*. Питание должно подаваться через разъем. В качестве языка программирования используется язык Си.

Перед программированием микроконтроллера была произведена настройка выводов *STM32* в соответствии со схемой, были заданы параметры чтения тактового сигнала для высокой и низкой частоты, поскольку у микроконтроллера на плате имеются 2 резонатора на 8 МГц и 32,768 кГц, это нужно указать в среде программирования. Таким образом микроконтроллер при запуске начнет руководствоваться не внешним сигналом, а генерировать его сам и руководствоваться возвратным сигналом с резонаторов.

На рисунке 1.18 изображен среда разработке *STM32CubeIDE*.

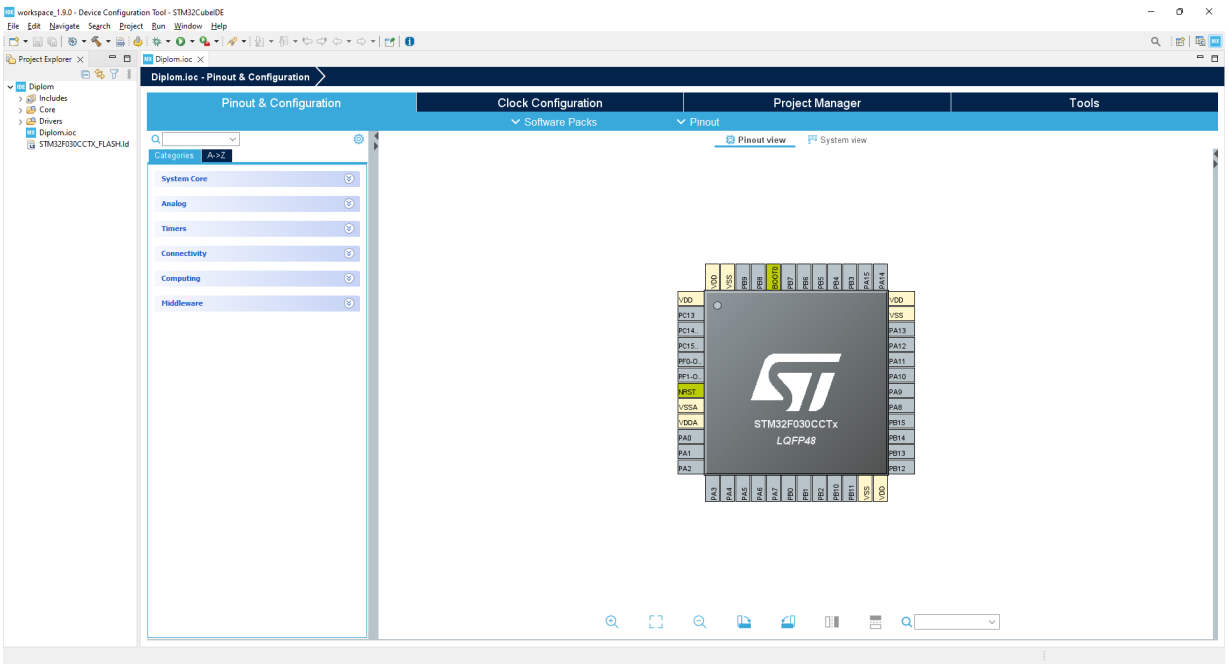


Рисунок 1.18 – Среда разработки *STM32CubeIDE*

Так же необходимо настроить резонатор см. Рисунок 1.19.

Для сигнальных светодиодов был включен режим *pulldown*. Для передачи данных будет использоваться *USART1*(*PA9* и *PA10*, его также необходимо включить. Для *SYS* был включен режим отладки (Разъемы *PA13* и *PA14*). *I2C* так же был подключен (*PB10* и *PB11*) для вывода на дисплей. На рисунке 1.20 можно увидеть микроконтроллер с заданными параметрами.

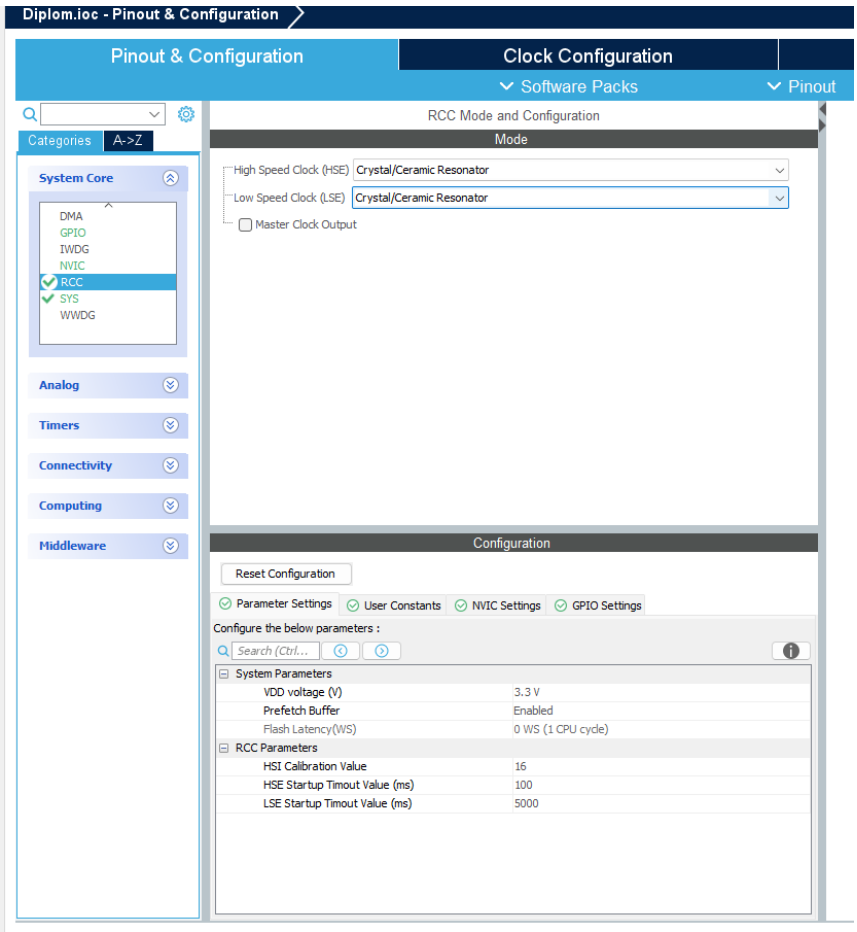


Рисунок 1.19 – Настройка резонаторов

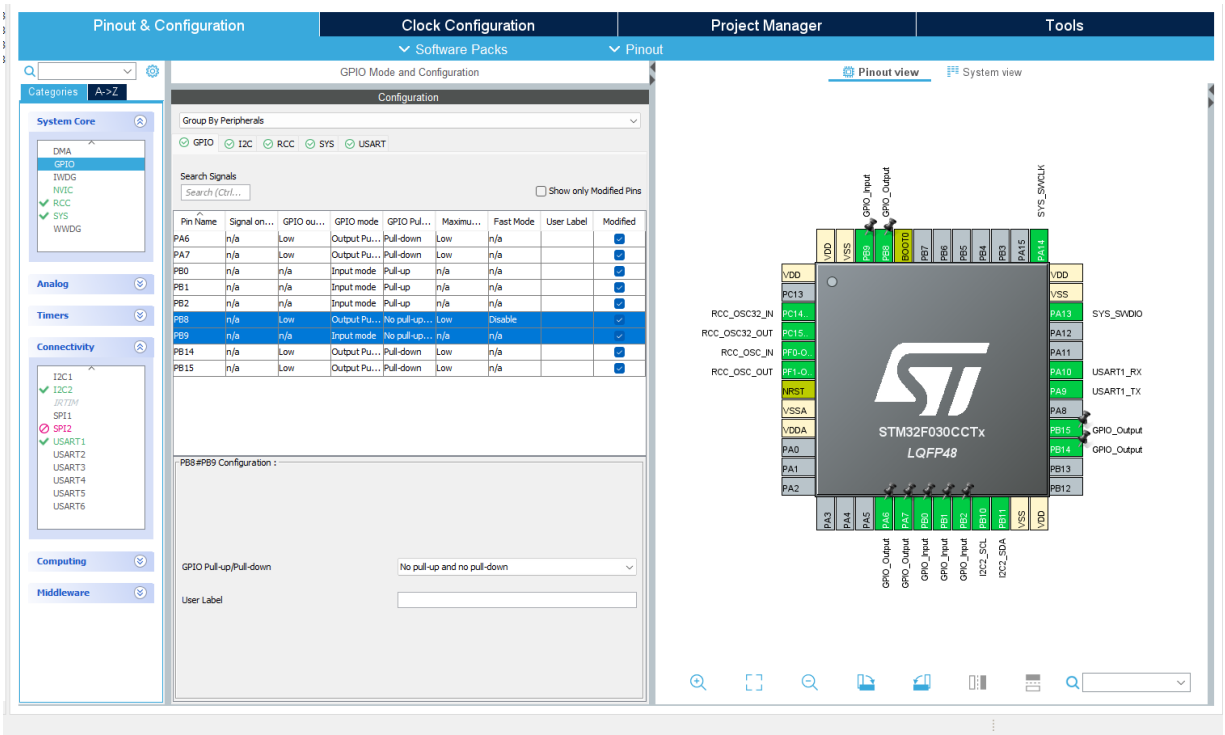


Рисунок 1.20 – Параметры контактов

1.7.2 Написание программного обеспечения

Блок схема алгоритма программы представлена на рисунке 1.21.

После запуска устройства открывается главное меню с предложением выбрать режим работу. Управление интерфейсом устройства осуществляется за счет комбинаций кнопок (*SW2*…*SW4*). Движения вверх и вниз по списку режимов осуществляется за счет нажатия на кнопки *SW2*, *SW3*; для подтверждения режима достаточно нажать на кнопку *SW4*. Для возврата в предыдущее меню вместе нажать *SW2* и *SW4*.

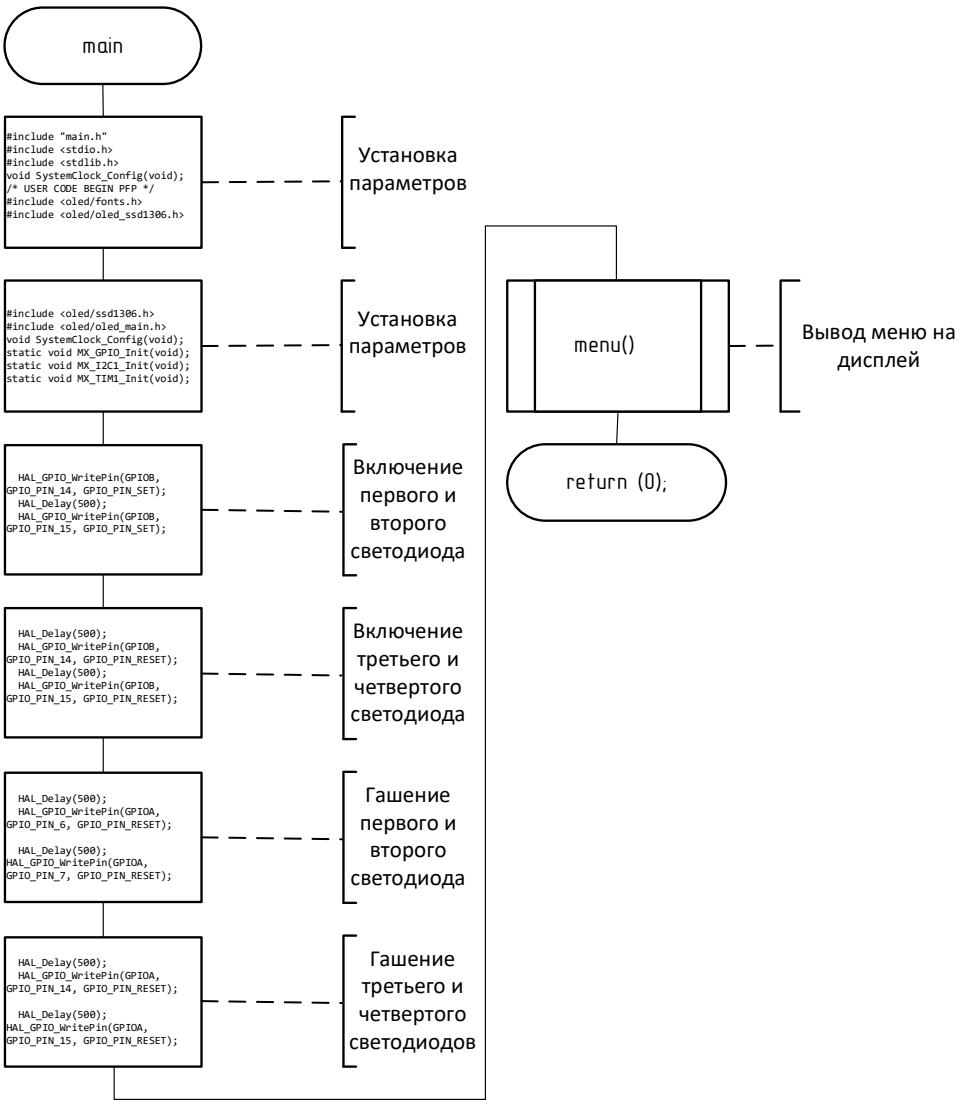


Рисунок 1.21 – Схема программы

Основные режимы работы программы:

- режим чтения позволяет копировать и сохранять в память уже записанные копии ключей;

- режим записи позволяет выбирать отдельный ключ из памяти и программировать ими ключ без записанной на него информации;

- режим эмуляции позволяет выбрать отдельный ключ из памяти или перебрать все ключи в памяти по очереди и воспроизвести их.

1.7.3 Принцип работы *I2C*

Модуль *I2C* у *STM32* обладает следующими особенностями:

- может работать в двух режимах *Fm(fast mode)* и *Sm(standart mode)*; первый работает на частотах до 400 кГц, второй до 100 кГц;

- буфер размером 1 байт с поддержкой *DMA*;

- поддерживает аппаратный подсчет контрольной суммы;

- на его основе возможна реализуется *SMBus* (*System Management Bus*) и *PMBus*(*Power Management Bus*);

- два вектора прерывания, генерируются при успешной передаче и при возникновении ошибки;

- фильтр для борьбы с шумами;

- может работать в режиме *Master* или *Slave*.

в режиме *Master*:

- генерирует тактирующий сигнал;

- генерирует *START* и *STOP*.

В режиме *Slave*:

- можно программировать основной и альтернативный адрес на который он будет отзываться;

- определяет *STOP*.

Для работы с *OLED*-дисплеем была использована библиотека *SSD*1306, так как был выбран дисплей, основанный на базе одноименного чипа.

SSD1306 способен работать по нескольким протоколам, в том числе

по *SPI* и *I2C*. *I2C* используется при ограниченном количестве доступных контактов. *SPI* требует больше проводов, но позволяет передавать данные намного быстрее.

1.7.4 Структура ключа *RFID*

Структура ключа *RFID* представлена на рисунке 1.22.

1	1	1	1	1	1	1	1	1	9 bit header bits, all 1's
8 bit version number or customer ID.				D00	D01	D02	D03	P0	
				D04	D05	D06	D07	P1	
32 Data Bits				D08	D09	D10	D11	P2	Each group of 4 bits is followed by an Even parity bit
				D12	D13	D14	D15	P3	
				D16	D17	D18	D19	P4	
				D20	D21	D22	D23	P5	
				D24	D25	D26	D27	P6	
				D28	D29	D30	D31	P7	
				D32	D33	D34	D35	P8	
				D36	D37	D38	D39	P9	
4 column Parity bits				PC0	PC1	PC2	PC3	S0	1 stop bit (0)

Рисунок 1.22 – Структура ключа *RFID*

Первые 9 бит состоят из логических единиц. Эти биты используются в качестве маркерной последовательности для обозначения начала строки. Далее следуют 10 групп по 4 бита данных и 1 бит четности. Наконец, идет 4 бита четности столбца (четные) и стоповый бит (0).

Существует 3 основные модели модуляции ключа.

При манчестерской схеме кодирования (см. рисунок 1.23), переход между битами осуществляется за счет периода соответствующего уровня, то есть для перехода на ноль берется период высокого уровня и наоборот.

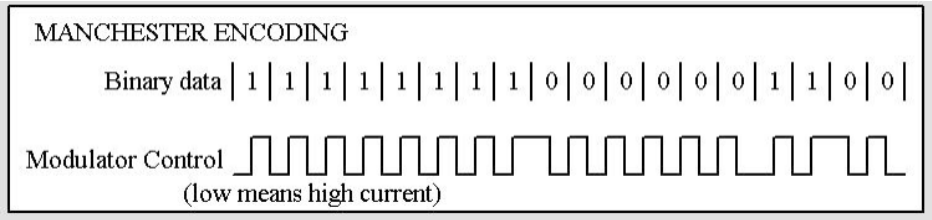


Рисунок 1.23 – Манчестерская схема кодирования

Двухфазная схема кодирования заключается в кодирование единице периодом нуля (см. рисунок 1.24).

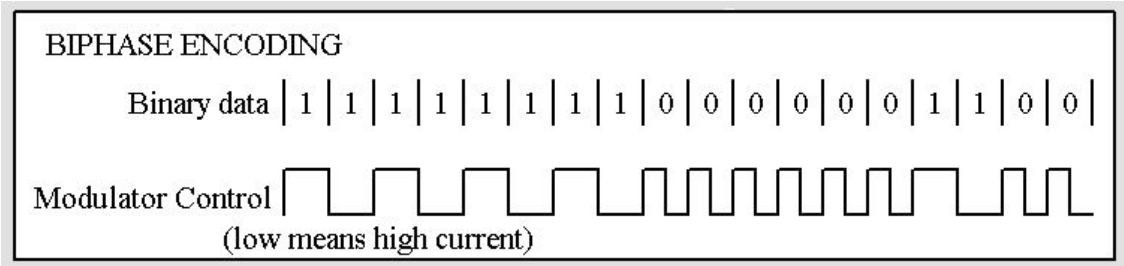


Рисунок 1.24 – Двухфазная схема кодирования

PSK-кодирование – радиочастотное поле модулируется таким образом, что с каждым тактовым периодом происходит переход. Это означает, что может быть до 64, 32 или 16 переходов на бит в зависимости от того, какую битовую длину использует тег. То есть, кодирование происходит в зависимости от выбранной битовой длины. Для кодирования единицы достаточно просто отсчитать выбранную длину, но для кодирования нуля нужно увеличить длину высокого уровня сигнала вдвое (см. рисунок 1.25).

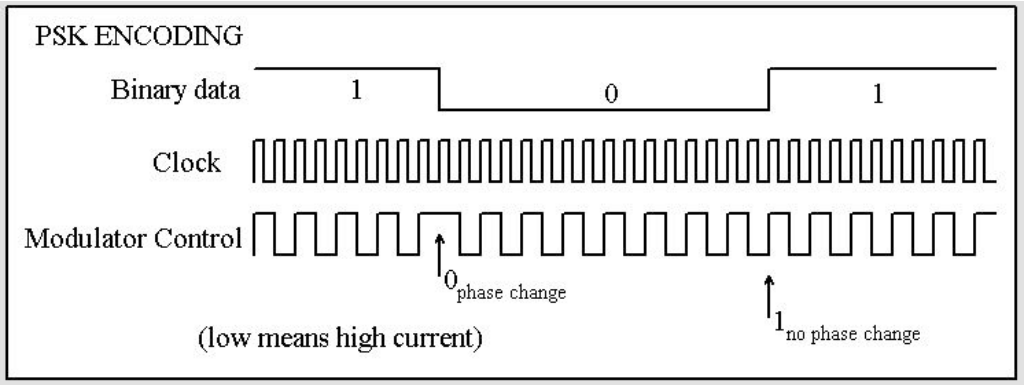


Рисунок 1.25 – *PSK*-кодирование

2 Охрана труда

Этот раздел посвящен технике безопасности при работе с вычислительными машинами и комплексами, а также другой аппаратуры участвующей в разработке универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*.

2.1 Нормы освяященности рабочей зоны

2.1.1 Нормы освещённости при работе за компьютером

К рабочему месту оператора электронной вычислительной машины выдвигаются такие же требования, как и к обычным рабочим местам по освещённости рабочей зоны. Освещённость должна быть 350 лк.

2.1.2 Нормы освещённости при работе с радиоэлементами

Поскольку монтаж радиодеталей требует наивысшей точности то и к освящению рабочей зоны монтажника выдвигаются соответствующие требования по освещённости. Освещённость рабочей зоны должна варьироваться от 2000 до 5000 лк.

2.2 Правила безопасности при работе с ВТ

При длительной работе с вычислительной техникой стоит помнить о следующих правилах.

2.2.1 Меры безопасности перед началом работы за ВТ

Перед началом работы за ВТ следует:

- проверить исправность элементов питания системы;
- убедиться, что устройство, за которым будет проходить работа заземлено;
- проверить работоспособность устройства.

2.2.2 Техника безопасности во время работы за ВТ

При работе с устройством запрещается:

- находится с жидкостью или пищей возле ВТ;
- класть на корпус или дисплей компьютера посторонние предметы;

- осуществлять чистку корпуса во время работы ВТ;

- нарушать инструкции, установленные производителем компьютера.

2.2.3 Техника безопасности по окончанию работы за ВТ

Завершать работу следует руководствуясь следующими правилами:

- выключать ВТ следует по алгоритму предусмотренным производителем;

- обесточить подключенные устройства;

- убедиться в том, что устройство завершило работу;

- привести рабочее пространство к состоянию, в котором оно было до начала работы.

2.3 Правила безопасности при монтаже радиоэлементов

2.3.1 Меры безопасности перед началом монтажа

Перед началом монтажных работ следует:

- убедиться, что плата, на которую будет осуществляется монтаж не подключена к сети;

- проверить исправность прибора, которым будет осуществляться монтаж;

- убедиться в том, что место контакта с рукой монтажника заизолировано.

2.3.2 Меры безопасности во время монтажа:

Работа с мелкими радиодеталями должна осуществляться под микроскопом или же под увеличительным стеклом достаточной кратности, для того чтобы не перенапрягать зрение.

Во время пайки нужно учитывать следующие:

- нельзя касаться нагревательной части работающего паяльника, термофена.

- весь монтаж должен осуществляться при помощи пинцета, это предотвратит ожоги пальцев, механическое повреждения компонентов платы, повреждение компонентов за счет воздействия на них статического электричества;

- при использование свинцово-содержащих припоев и припоев таких как оловянно-спиртовой раствор, хлористый цинк, необходимо позаботится о защите дыхательных путей, так при нагреве этих припоев и флюса выделяются токсич-

ные пары.

2.3.3 Меры безопасности после завершения монтажа

После завершения работу следует:

- убедиться в правильности монтажа компонентов;
- убедиться в надежности крепления контактов компонентов;
- убедиться в отсутствии ненужных соединений контактов;
- отключить устройство от монтажа от сети;
- отмыть плату от флюса.

Заключение

В ходе выполнения выпускной квалификационной работы был разработан универсальный ключ идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*.

В ходе данной реализации данной работы были изучены методы разработки и проектирования печатных плат.

Были разработаны схемы: структурная схема и схема электрическая принципиальная для универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*, схема программы для универсального ключа идентификации и контроля *RFID*-меток на основе микроконтроллера *STM32*.

Разработанное устройство предназначено для копирования, хранения и воспроизведения *RFID* меток на частоте 125 кГц.

Так же были получены навыки работы в среде разработки и проектирования *PADS Professional*.

Список использованных источников

1. Жеребцов И. П. Основы электроники. Изд. 5-е, — Л.: 1989. — С. 120—121.

2. Документация к STM32 [Электронный ресурс] / Режим доступа: https://st.com свободный. (Дата обращения 14.04.2022)

3. Хорвиц П., Хилл У. Искусство схемотехники. - М: Мир, 1993.

4. Рекомендации к трассировки печатных плат [Электронный ресурс] / Режим доступа http://microsin.net/adminstuff/others/pcb-design-guidelines-needs-to-know.html свободный (Дата обращения 3.05.2022)

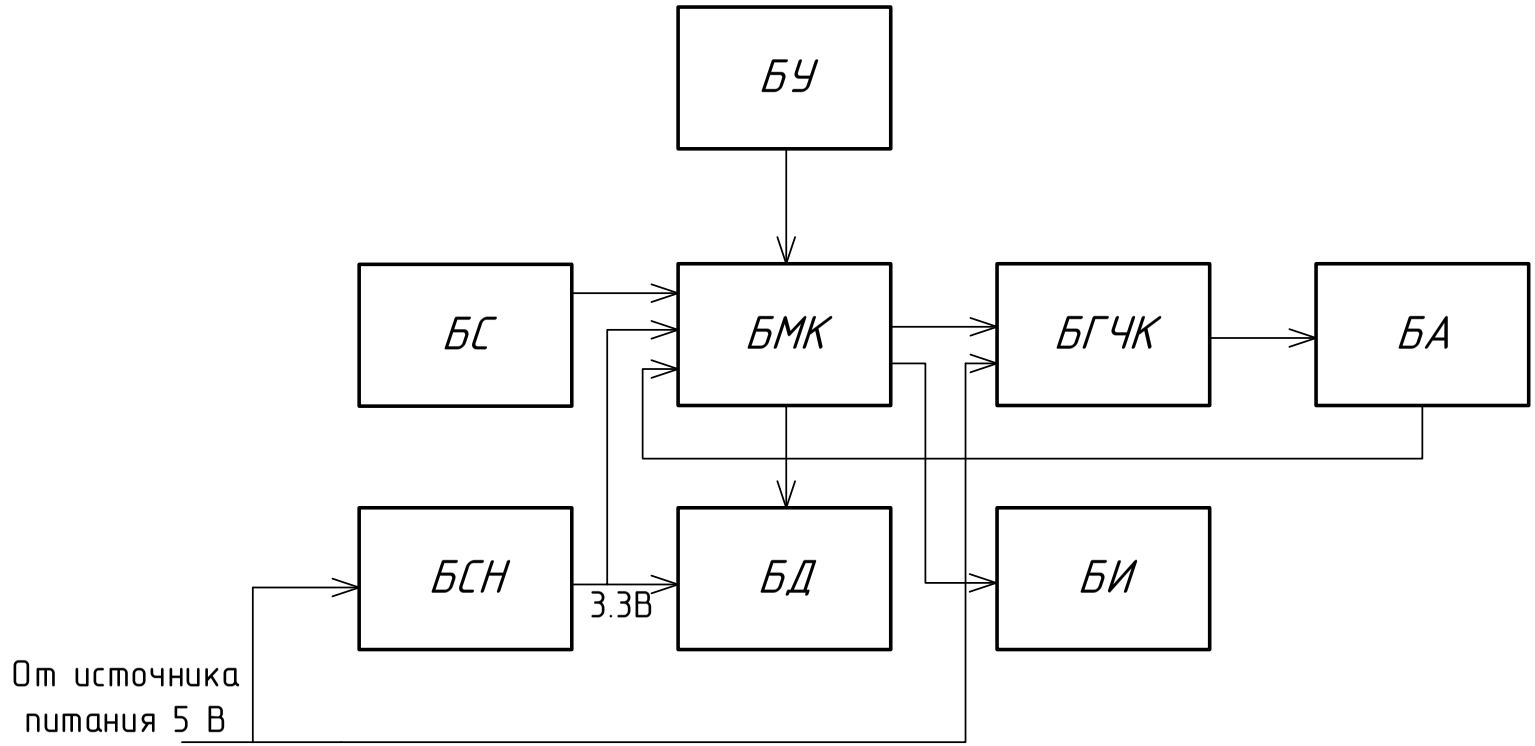
5. Пособие по программирование в среде STM32CubeMX [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/310742/ свободный. (Дата обращения 11.05.2022)

Приложение А1

(Схема электрическая структурная)

Инв.№ подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм. | Лист | №докум. | Подп. | Дата



БС – Блок синхронизации
БСН – Блок стабилизации напряжения
БУ – Блок управления
БМК – Блок микроконтроллера
БД – Блок дисплея
БГЧК – Блок генерации/чтения ключа
БИ – Блок идентификации
БА – Блок антенны

								ПГУ 1.090301.02.002 Э1	
Изм.	Лист	N докум.	Подп.	Дата					
Разраб.		Ампилов Д.О.			Универсальный ключ идентификации и контроля КПО-метки на основе микроконтроллера STM32 Схема электрическая структурная		Лит.	Масса	Масштаб
Пров.		Бычков А.С.							
Т.контр.							Лист	Листов 1	
Реценз.									
Н.контр.		Бычков А.С.						Гр. 18ВВ1	
Утв.		Микрюкин М.А.							

ПГУ 1.090301.02.001 ПЗ

Лист 52

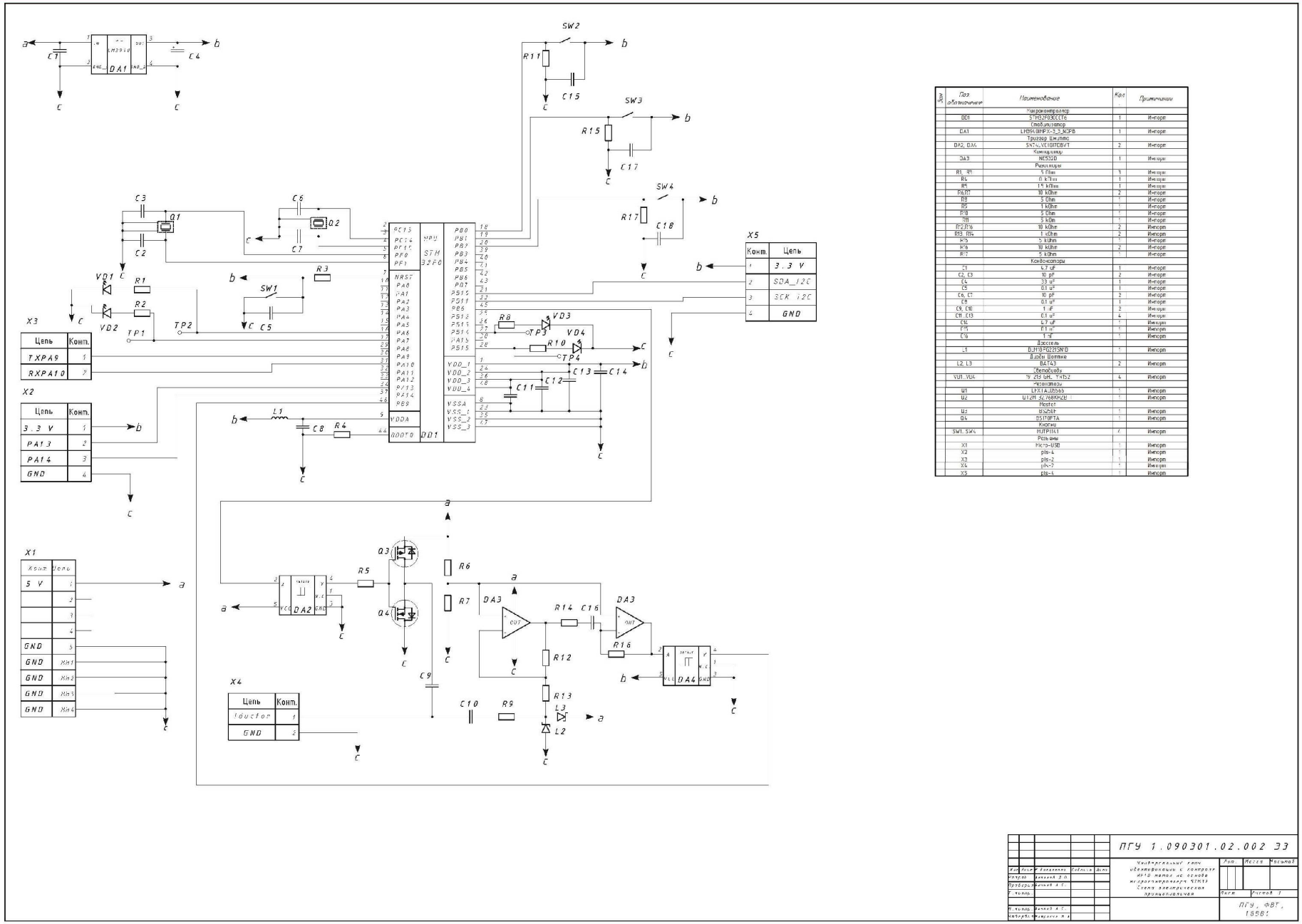
Приложение А2

(Схема электрическая принциальная)

Инв.№ подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм. | Лист | №докум. | Подп. | Дата

ПГУ 1.090301.02.001 ПЗ



Поз. обозначение	Наименование	Кол	Примечание
	Микроконтроллер		
DD1	STM32F030CCT6	1	Интерэл
	Стабилизатор		
DA1	LM3940IMP-3.3_NOPB	1	Интерэл
	Триггер Шмитта		
DA2, DA4	SN74LVC1G17DBVT	2	Интерэл
	Компаратор		
DA3	NE5532D	1	Интерэл
	Резисторы		
R1, R4	5.1Ohm	2	Интерэл
R2	6.1kOhm	1	Интерэл
R3	1.5 kOhm	1	Интерэл
R5,R7	10 kOhm	2	Интерэл
R8	5 Ohm	1	Интерэл
R9	1 kOhm	1	Интерэл
R10	5 Ohm	1	Интерэл
R11	5 kOm	1	Интерэл
R12,R16	10 kOhm	2	Интерэл
R13, R14	1 kOhm	2	Интерэл
R15	5 kOhm	1	Интерэл
R16	10 kOhm	2	Интерэл
R17	5 kOhm	1	Интерэл
	Конденсаторы		
C1	4.7 uF	1	Интерэл
C2, C3	10 pF	2	Интерэл
C4	33 uF	1	Интерэл
C5	0.1 uF	1	Интерэл
C6, C7	10 pF	2	Интерэл
C8	0.1 uF	1	Интерэл
C9, C10	1 uF	2	Интерэл
C11..C13	0.1 uF	4	Интерэл
C14	4.7 uF	1	Интерэл
C15	0.1 uF	1	Интерэл
C16	1 nF	1	Интерэл
	Диоссель		
L1	BLM18PG221SN1D	1	Интерэл
	Дроссель Шоттки		
L2, L3	BAT43	2	Интерэл
	Светодиоды		
VD1..VD4	Yellow, 1.2V, 1mA	4	Интерэл
	Генераторы		
U1	LFXTAL025355	1	Интерэл
U2	ULPM-SG7.68MHZEH	1	Интерэл
	Монтаж		
U3	BS250	1	Интерэл
Q4	BSS138TA	1	Интерэл
	Кнопки		
SW1..SW4	MJTP1151	4	Интерэл
	Разъемы		
X1	Micro-USB	1	Интерэл
X2	pls-4	1	Интерэл
X3	pls-2	1	Интерэл
X4	pls-2	1	Интерэл
X5	pls-4	1	Интерэл

					ПГУ 1.090301.02.002 ЭЗ	
Разраб.	Ваняшкин			Универсальный канал обеспечивающий и контролят ИКТД метод на основе инфракрасного канала. Схема электрическая принципиальная	Лит.	Масса	Масштаб
Провер.	Данилов Э.Л.						
Преподав.	Данилов Э.Л.				Лист	Листов 1	
Т.контр.							
Н.контр.	Данилов А.Г.				ПГУ, ФВТ, 1558т		
Утвердил	Ваняшкин А.В.						

Приложение А3

(Обязательное)

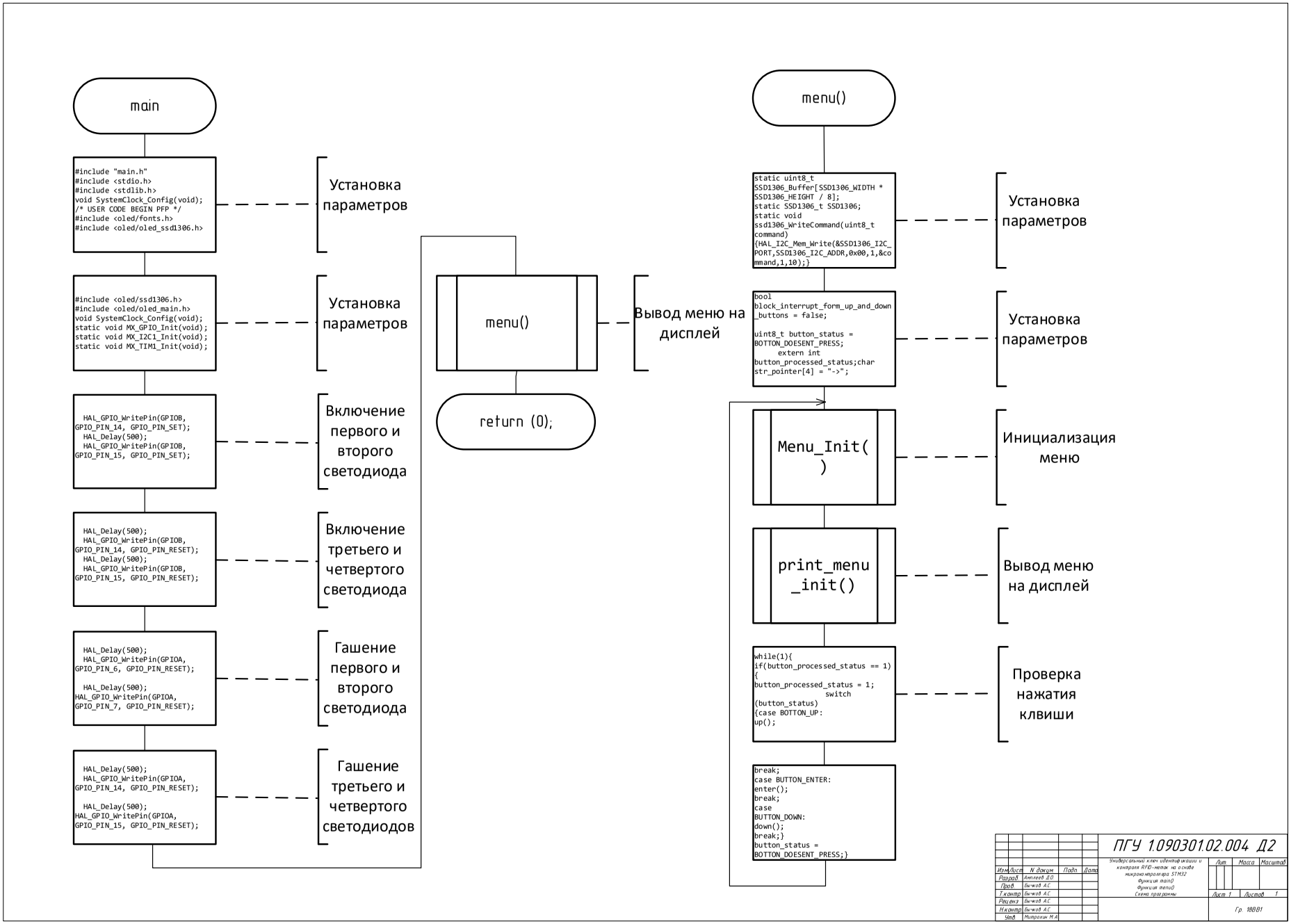
Схема программы

Инв.№ подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм. | Лист | №докум. | Подп. | Дата

ПГУ 1.090301.02.001 ПЗ

Лист 56



ПГУ 1.090301.02.004 Д2

Универсальный ключ идентификации и контроля RFID-меток на основе микроконтроллера STM32

Функция main()
Функция menu()
Схема программы

Изм.	Лист	N докум.	Подп.	Дата
Разраб.	Аминаев Д.О.			
Пров.	Бычков А.С.			
Т.контр.	Бычков А.С.			
Реценз.	Бычков А.С.			
Н.контр.	Бычков А.С.			
Утв.	Митрохин М.А.			

Лит.	Масса	Масштаб
Лист 1	Листов 1	

Гр. 18ВВ1

Инв.№ подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм. | Лист | №докум. | Подп. | Дата

ПГУ 1.090301.02.001 ПЗ

Лист 57

print_menu_init()

```
static uint8_t
SSD1306_Buffer[SSD1306_WIDTH *
SSD1306_HEIGHT / 8];
static SSD1306_t SSD1306;
static void
ssd1306_WriteCommand(uint8_t
command)MenuItem_t *

currentItem_buff = currentItem;
```

Установка параметров

```
print_rectangle_on_head();
char str[20] = ">> Выбор режима работы
<<";
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10,
White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));
{
HAL_I2C_Mem_Write(&SSD1306_I2C_PORT,

SSD1306_I2C_ADDR,0x00,1,&command,1,10);}
```

Печать Выбранное название

```
for (uint8_t row = first_menu_row;
row <= fourth_menu_row; row = row +
row_step)
      {
      if(row == first_menu_row)
      {
      char str_pointer[4] = "->";
      ssd1306_SetCursor(0, row);
      ssd1306_WriteString(str_pointer
,  Font_7x10, White);}
```

Печать указателя на меню (сверху)

```
itoa(currentItem_buff -> id,
str, 10);
ssd1306_SetCursor(start_print_id
_menu_x, row);
ssd1306_WriteString(str,
Font_7x10, White);
memset(str, 0, sizeof(str));
```

Вывод номера пункта меню

```
strncpy(str, currentItem_buff ->
name, 15);
ssd1306_SetCursor(start_print_name_me
nu_x, row);
ssd1306_WriteString(str,  Font_7x10,
White);
ssd1306_UpdateScreen();
currentItem_buff = currentItem_buff -
> down;
```

Печать пункта меню

```
if(currentItem_  buff
== 0)
    {
    break;
    }}
```

НЕТ

Да

Break;

scroll_bar( );

Menu_Init()

```
void (*p_print_rows_on_oled_if_up) (void);
      p_print_rows_on_oled_if_up =
print_rows_on_oled_if_up;
void (*p_print_rows_on_oled_if_down) (void);
      p_print_rows_on_oled_if_down =
print_rows_on_oled_if_down;
void (*p_return_from_menu)(void);
p_return_from_menu = return_from_menu;
void (*p_action) (void);
p_action = action;
```

Создание и сохранение функции

```
void (*p_print_rows_on_oled_if_up) (void);
p_print_rows_on_oled_if_up =
print_rows_on_oled_if_up;
void (*p_print_rows_on_oled_if_down) (void);
p_print_rows_on_oled_if_down =
print_rows_on_oled_if_down;
void (*p_return_from_menu)(void);
p_return_from_menu = return_from_menu;
void (*p_action) (void);
p_action = action;
```

Инициализация функции первого меню

```
void (*p_do_it_function_menu_2)
(void);
p_do_it_function_menu_2 =
do_it_function_menu_2;
void (*p_items_menu_2_set_par_1)
(void);
p_items_menu_2_set_par_1 =
items_menu_2_set_par_1;
```

Инициализация функции второго меню

```
void
(*p_do_it_function_menu_3
) (void);

p_do_it_function_menu_3 =
do_it_function_menu_3;
```

Инициализация функции третьего меню

```
items[0].up = 0;
items[0].down =
&items[1];
items[0].child =
&items_menu_1[0];
items[0].parent = 0;
items[0].id = 1;
```

Заполняем списки первого меню

```
items[0].name = "Menu_1";
items[0].updateScreen_up =
p_print_rows_on_oled_if_up;
items[0].updateScreen_down =
p_print_rows_on_oled_if_down;
items[0].makeAction = 0;
      …
```

```
      ...
items[6].up = &items[5];
      items[6].down = 0;
      items[6].child = 0;
      items[6].parent = 0;
      items[6].id = 7;
      items[6].name = "Menu_7";
      items[6].updateScreen_up =
p_print_rows_on_oled_if_up;
      items[6].updateScreen_down =
p_print_rows_on_oled_if_down;
      items[6].makeAction  = 0;
```

Заполняем списки первого меню

```
items_menu_1[0].up = 0;
items_menu_1[0].down =
&items_menu_1[1];
items_menu_1[0].id = 1;
items_menu_1[0].child = 0;
items_menu_1[0].parent =
&items[0];
items_menu_1[0].name = "set par
1";
```

Создаем и заполняем следующее меню

```
items_menu_1[0].updateScreen_up
= p_print_rows_on_oled_if_up;
items_menu_1[0].updateScreen_dow
n =
p_print_rows_on_oled_if_down;
items_menu_1[0].makeAction =
p_items_menu_1_set_par_1;
```

Создаем и заполняем следующее меню

```
items_menu_4[9].up =
&items_menu_4[8];
items_menu_4[9].down = 0;
items_menu_4[9].child = 0;
items_menu_4[9].parent =
&items[3];
items_menu_4[9].id = 10;
items_menu_4[9].name = "Выход";
```

Заполнение последнего меню

```
items_menu_4[9].updateScreen_up
= p_print_rows_on_oled_if_up;
items_menu_4[9].updateScreen_dow
n =
p_print_rows_on_oled_if_down;
items_menu_4[9].makeAction =
p_return_from_menu;
```

Заполнение последнего меню

Return 0;

Изм.	Лист	N докум.	Подп.	Дата	ПГУ 1.090301.02.005 Д2		
Разраб.	Аметлей Д.О.				Универсальный ключ идентификации и контроля RFID-меток на основе микроконтроллера STM32	Лит.	Масса	Масштаб
Пров.	Быков А.С.				Функции print_menu_init(), Menu_Init()			
Т.контр.					Схемы программ	Лист 1	Листов 1	
Реценз.						Гр. 18ВВ1		
Н.контр.	Быков А.С.							
Утв.	Жигорькин М.А.							

Приложение Б

(Обязательное)

Универсальный ключа идентификации и контроля RFID-меток на основе микроконтроллера STM32

Листинг программы

Файл Src/menu/ oled_simulation_menu.c

```
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

#include <oled/fonts.h>
#include <oled/oled_ssd1306.h>
#include <oled/ssd1306.h>
#include <oled/oled_main.h>

#include <menu/oled_simulation_menu.h>

// Main struct menu
typedef struct Struct{
 struct Struct* up;       // pointer on up element of list
 struct Struct* down;     // pointer on down element of list
 struct Struct* child;     // Pointer on child menu
 struct Struct* parent;     // Pointer on parent menu

 uint8_t id;        // Number of menu
 char *name;        // Name menu
 void ( *updateScreen_up ) (void );  // Function print graphic on OLED
 void ( *updateScreen_down ) (void );  // Function print graphic on OLED
 void ( *makeAction) ( void );
}MenuItem_t;

// How many menu items in menu
#define MENU_ITEM_NUM 7
#define MENU_1_ITEM_NUM 5
#define MENU_2_ITEM_NUM 3
#define MENU_3_ITEM_NUM 2
#define MENU_4_ITEM_NUM 10
// Create menu item array structure for all menus
MenuItem_t items[MENU_ITEM_NUM];
MenuItem_t items_menu_1[MENU_1_ITEM_NUM];
MenuItem_t items_menu_2[MENU_2_ITEM_NUM];
MenuItem_t items_menu_3[MENU_3_ITEM_NUM];
```

```
MenuItem_t items_menu_4[MENU_4_ITEM_NUM];
MenuItem_t * currentItem = &items[0];  // Create and set pointer on first
element of list
bool block_interrupt_form_up_and_down_buttons = false;   // Flag for lock
interrupt from 'up' and 'down' buttons in some cases
uint8_t button_status = BOTTON_DOESENT_PRESS;     // Pressed button
extern int button_processed_status;   // For interrupt work only one time
char str_pointer[4] = "->";     // How look pointer on menu item
// Rows coordinates
uint16_t first_menu_row = 16;
uint16_t second_menu_row = 28;
uint16_t third_menu_row = 40;
uint16_t fourth_menu_row = 52;
uint8_t row_step = 12;
uint16_t start_print_id_menu_x = 15;
uint16_t start_print_name_menu_x = 30;
// ------------------------------------------------------------------------
----------------
/*
This function print scrollbar on right part of OLED.

 */
void scroll_bar(void)
{
 // Work size scrollbar
 uint16_t start_x_scrollbar = 124;
 uint16_t start_y_scrollbar = 17;
 uint16_t active_width = 3;
 uint16_t lenght_all_scrollbar = 48;

 uint8_t menu_items_counter = 1;

 MenuItem_t * currentItem_buff = currentItem;

 if ((currentItem_buff -> up) != 0)    // Шаг вверх, если существует
 {
  do{
   currentItem_buff = currentItem_buff -> up;   // Переход в начало меню
```

```
 }
 while ((currentItem_buff -> up) != 0);     // Если вершина элемента не
найдена
 }

 if ((currentItem_buff -> up) == 0)      // Если найден в верхней части
пункта меню
 {
  do{
   currentItem_buff = currentItem_buff -> down;
   menu_items_counter++;       // Подсчитываем количество списков в меню
  }
  while ((currentItem_buff -> down) != 0);
 }

 ssd1306FillRect(start_x_scrollbar, start_y_scrollbar, active_width,
lenght_all_scrollbar - 2, Black);


 ssd1306_DrawRectangle(start_x_scrollbar - 1, start_y_scrollbar - 1,
start_x_scrollbar + active_width, 63, White);
 ssd1306_UpdateScreen();

 // Печать строки прокрутки
 int id_for_line = currentItem -> id;
 // Print scroling line
 uint16_t line_lenght = (lenght_all_scrollbar/menu_items_counter + 1);
 uint16_t start_lenght = 16 + ((id_for_line - 1)*line_lenght);
 // Печатаем активную часть полосы прокрутки (строку)
 ssd1306FillRect(start_x_scrollbar, start_lenght, active_width,
line_lenght, White);

 ssd1306_UpdateScreen();
 }
 // ------------------------------------------------------------------------
 ----------------
 void print_rectangle_on_head(void)
 {
```

```
 ssd1306_DrawRectangle(0, 0, 127 , 15, White);
 ssd1306_UpdateScreen();
}
// ------------------------------------------------------------------------
----------------
// Очистить некоторые пункты меню.
void clear_menu_items (bool first, bool second, bool third, bool fourth)
{
uint8_t start_row_x = 15;
 char str[16] = "               ";   // Должно быть 15

 if(first == true)
 {
  ssd1306_SetCursor(start_row_x, first_menu_row);
  ssd1306_WriteString(str,  Font_7x10, White);
 }
 if(second == true)
 {
  ssd1306_SetCursor(start_row_x, second_menu_row);
  ssd1306_WriteString(str,  Font_7x10, White);
 }
 if(first == true)
 {
  ssd1306_SetCursor(start_row_x, third_menu_row);
  ssd1306_WriteString(str,  Font_7x10, White);
 }
 if(fourth == true)
 {
  ssd1306_SetCursor(start_row_x, fourth_menu_row);
  ssd1306_WriteString(str,  Font_7x10, White);
 }
 ssd1306_UpdateScreen();
}
// ------------------------------------------------------------------------
----------------
void print_rows_on_oled_if_up(void)
{
 char str[16] = {0};
```

```
clear_menu_items (true , true , true , true );

print_rectangle_on_head();

// Печать указателя на первый пункт меню
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str_pointer,  Font_7x10, White);

MenuItem_t * currentItem_buff_up = currentItem;   // Создать буфер по выбранному текущему указателю элемента.
for (uint8_t row = first_menu_row; row <= fourth_menu_row; row = row + row_step)
{
 // Выводим номер пункта меню
 itoa(currentItem_buff_up -> id, str, 10);
 ssd1306_SetCursor(start_print_id_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);

 // Выводим название пункта меню
 strncpy(str, currentItem_buff_up -> name, 15);
 ssd1306_SetCursor(start_print_name_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);

 currentItem_buff_up = currentItem_buff_up -> down;  // Делаем шаг вниз

 // Печатаем только существующие элементы
 if(currentItem_buff_up == 0)      // Если нет следующего элемента
 {
  break;
 }
}
ssd1306_UpdateScreen();
scroll_bar();
}
// ------------------------------------------------------------------------
----------------
void print_rows_on_oled_if_down(void) // Пункт меню печати текста
```

```
{
char str[16] = {0};

clear_menu_items (true , true , true , true );

print_rectangle_on_head();

// Печать указателя на первый пункт меню
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str_pointer,  Font_7x10, White);

MenuItem_t * currentItem_buff = currentItem;
for (uint8_t row = first_menu_row; row <= fourth_menu_row; row = row +
row_step)
 {
  // Выводим номер пункта меню
        itoa(currentItem_buff -> id, str, 10);
  ssd1306_SetCursor(start_print_id_menu_x, row);
  ssd1306_WriteString(str,  Font_7x10, White);

  // Выводим название пункта меню
  memset(str, 0, sizeof(str));
     strncpy(str, currentItem_buff -> name, 15);
     ssd1306_SetCursor(start_print_name_menu_x, row);
     ssd1306_WriteString(str,  Font_7x10, White);

     currentItem_buff = currentItem_buff -> down;   // Делаем шаг вниз

     // Печатаем только существующие элементы
     if(currentItem_buff == 0)       // Если нет следующего элемента
     {
     break;
     }
  }
  ssd1306_UpdateScreen();
  scroll_bar();
}
```

```
// ---------------------------------------------------------------------------
---------------
// Печатаем первые 4 пункта меню (только в первый раз)
void print_menu_init(void)
{
MenuItem_t * currentItem_buff = currentItem;

print_rectangle_on_head();

//Распечатать выбранное название меню
char str[20] = ">> Выбор режима работы <<";
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

for (uint8_t row = first_menu_row; row <= fourth_menu_row; row = row +
row_step)
{
 if(row == first_menu_row)
 {
  // Печать указателя на меню (сверху)
  char str_pointer[4] = "->";
  ssd1306_SetCursor(0, row);
  ssd1306_WriteString(str_pointer,  Font_7x10, White);
 }
 // Выводим номер пункта меню
 itoa(currentItem_buff -> id, str, 10);
 ssd1306_SetCursor(start_print_id_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);
 memset(str, 0, sizeof(str));

 // Печать меню пункта меню
 strncpy(str, currentItem_buff -> name, 15);
 ssd1306_SetCursor(start_print_name_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);

 ssd1306_UpdateScreen();
```

```
 currentItem_buff = currentItem_buff -> down;

 if(currentItem_buff == 0)      // Если нет пункта меню
 {
  break;
 }
}
scroll_bar();
}
// ------------------------------------------------------------------------
----------------
void action(void)
{
char str[16] = {0};

clearn_oled();

print_rectangle_on_head();

// Печатаем выбранное название меню поверх OLED
MenuItem_t * currentItem_buff_parent = currentItem;
currentItem_buff_parent = currentItem_buff_parent -> parent;

strncpy(str, currentItem_buff_parent -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

MenuItem_t * currentItem_buff = currentItem;

for (uint8_t row = first_menu_row; row <= fourth_menu_row; row = row +
row_step)
{
 if(row == first_menu_row)
 {
  // Печать указателя на меню
```

```
  char str_pointer[4] = "->";
  ssd1306_SetCursor(0, row);
  ssd1306_WriteString(str_pointer,  Font_7x10, White);
 }
 // Выводим номер пункта меню
 itoa(currentItem_buff -> id, str, 10);
 ssd1306_SetCursor(start_print_id_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);

 // Печать меню пункта меню
 strncpy(str, currentItem_buff -> name, 15);
 ssd1306_SetCursor(start_print_name_menu_x, row);
 ssd1306_WriteString(str,  Font_7x10, White);
 ssd1306_UpdateScreen();

 currentItem_buff = currentItem_buff -> down;
 if(currentItem_buff == 0)
 {
  break;
 }
}
scroll_bar();
}
// ------------------------------------------------------------------------
----------------
void return_from_menu(void)
{
 currentItem = &items[0];          // Перейти в главное меню
 clearn_oled();


 char str[20] = ">> Выберете режим работы <<";
 ssd1306_SetCursor(10, 3);
 ssd1306_WriteString(str,  Font_7x10, White);
 ssd1306_UpdateScreen();

 print_menu_init();           // Распечатать все стартовое меню
}
```

```
void do_it_function_menu_1(void)
{
clearn_oled();

// Печатаем выбранное название меню
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Режим чтения", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
 HAL_Delay(200);
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
 HAL_Delay(200);

}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Возврат к первому пункту текущего меню
currentItem = &items_menu_1[0];         // Установить глобальный указатель
на первое меню
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_1_set_par_1(void)
```

```
{
clearn_oled();

// Печатаем выбранное название меню
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 1", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_1[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_1_set_par_2(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
```

```
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 2", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_1[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_1_set_par_3(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 3", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
```

```
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_1[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
////////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////////
void do_it_function_menu_2(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Режим программирования", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
```

```
button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
 HAL_Delay(50);
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
 HAL_Delay(50);
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_2[0];         // Set global pointer on first menu
action();             // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_2_set_par_1(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 1", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_2[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////
/////////////////
void do_it_function_menu_3(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Генерация", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
 HAL_Delay(500);
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
```

```
 HAL_Delay(500);

}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_3[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
////////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////////
// ------------------------------------------------------------------------
----------------
void do_it_function_menu_4(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Режим эмуляции", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
```

```
 HAL_Delay(20);
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
 HAL_Delay(500);

}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_1(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 1", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;
```

```
// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_2(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 2", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_3(void)
```

```
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 3", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_4(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
```

```
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 4", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_5(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 5", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
```

```
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();             // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_6(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 6", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ------------------------------------------------------------------------
----------------
void items_menu_4_set_par_7(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 7", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
```

```
// ---------------------------------------------------------------------
---------------
void items_menu_4_set_par_8(void)
{
clearn_oled();

// Print selected name of menu
char str[16] = {0};
strncpy(str, currentItem -> name, 15);
ssd1306_SetCursor(10, 3);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();
memset(str, 0, sizeof(str));

strncpy(str, "Set parametr 8", sizeof(str));
ssd1306_SetCursor(0, first_menu_row);
ssd1306_WriteString(str,  Font_7x10, White);
ssd1306_UpdateScreen();

button_status = BOTTON_DOESENT_PRESS;
block_interrupt_form_up_and_down_buttons = true;
do{
 // Doing settings
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
}while (button_status != BUTTON_ENTER);
block_interrupt_form_up_and_down_buttons = false;

// Return to first item of current menu
currentItem = &items_menu_4[0];         // Set global pointer on first menu
action();              // Print items on OLED
}
// ---------------------------------------------------------------------
----------------
// ---------------------------------------------------------------------
----------------
void Menu_Init (void)
{
// Make pointers on funsctions
```

```
// Main functions
void (*p_print_rows_on_oled_if_up) (void);
p_print_rows_on_oled_if_up = print_rows_on_oled_if_up;

void (*p_print_rows_on_oled_if_down) (void);    // Create pointer on func-
tion
p_print_rows_on_oled_if_down = print_rows_on_oled_if_down; // Save func-
tion print on pointer print_p

void (*p_return_from_menu)(void);
p_return_from_menu = return_from_menu;

void (*p_action) (void);          // Create pointer on function
p_action = action;           // Save function action on pointer action_p

// items_menu_1 menu functions
void (*p_do_it_function_menu_1) (void);     // Function "Do it". Works when
select it
p_do_it_function_menu_1 = do_it_function_menu_1;

void (*p_items_menu_1_set_par_1) (void);
p_items_menu_1_set_par_1 = items_menu_1_set_par_1;

void (*p_items_menu_1_set_par_2) (void);
p_items_menu_1_set_par_2 = items_menu_1_set_par_2;

void (*p_items_menu_1_set_par_3) (void);
p_items_menu_1_set_par_3 = items_menu_1_set_par_3;

// items_menu_2 menu functions
void (*p_do_it_function_menu_2) (void);     // Function "Do it". Works when
select it
p_do_it_function_menu_2 = do_it_function_menu_2;

void (*p_items_menu_2_set_par_1) (void);
p_items_menu_2_set_par_1 = items_menu_2_set_par_1;

// items_menu_3 menu functions
```

```
void (*p_do_it_function_menu_3) (void);     // Function "Do it". Works when
select it
p_do_it_function_menu_3 = do_it_function_menu_3;

// items_menu_4 menu functions
void (*p_items_menu_4_set_par_1) (void);
p_items_menu_4_set_par_1 = items_menu_4_set_par_1;

void (*p_items_menu_4_set_par_2) (void);
p_items_menu_4_set_par_2 = items_menu_4_set_par_2;

void (*p_items_menu_4_set_par_3) (void);
p_items_menu_4_set_par_3 = items_menu_4_set_par_3;

void (*p_items_menu_4_set_par_4) (void);
p_items_menu_4_set_par_4 = items_menu_4_set_par_4;

void (*p_items_menu_4_set_par_5) (void);
p_items_menu_4_set_par_5 = items_menu_4_set_par_5;

void (*p_items_menu_4_set_par_6) (void);
p_items_menu_4_set_par_6 = items_menu_4_set_par_6;

void (*p_items_menu_4_set_par_7) (void);
p_items_menu_4_set_par_7 = items_menu_4_set_par_7;

void (*p_items_menu_4_set_par_8) (void);
p_items_menu_4_set_par_8 = items_menu_4_set_par_8;

void (*p_do_it_function_menu_4) (void);     // Function "Do it". Works when
select it
p_do_it_function_menu_4 = do_it_function_menu_4;

// Fill in elements(nodes) of list (7 items)
// Main menu items
//////////////////////////////////////////////////////////////

items[0].up = 0;
```

```
items[0].down = &items[1];
items[0].child = &items_menu_1[0];
items[0].parent = 0;
items[0].id = 1;
items[0].name = "Menu_1";
items[0].updateScreen_up = p_print_rows_on_oled_if_up;
items[0].updateScreen_down = p_print_rows_on_oled_if_down;
items[0].makeAction = 0;

items[1].up = &items[0];
items[1].down = &items[2];
items[1].child = &items_menu_2[0];
items[1].parent = 0;
items[1].id = 2;
items[1].name = "Menu_2";
items[1].updateScreen_up = p_print_rows_on_oled_if_up;
items[1].updateScreen_down = p_print_rows_on_oled_if_down;
items[1].makeAction = 0;

items[2].up = &items[1];
items[2].down = &items[3];
items[2].child = &items_menu_3[0];
items[2].parent = 0;
items[2].id = 3;
items[2].name = "Menu_3";
items[2].updateScreen_up = p_print_rows_on_oled_if_up;
items[2].updateScreen_down = p_print_rows_on_oled_if_down;
items[2].makeAction = 0;

items[3].up = &items[2];
items[3].down = &items[4];
items[3].child = &items_menu_4[0];
items[3].parent = 0;
items[3].id = 4;
items[3].name = "Menu_4";
items[3].updateScreen_up = p_print_rows_on_oled_if_up;
items[3].updateScreen_down = p_print_rows_on_oled_if_down;
items[3].makeAction = 0;
```

```
items[4].up = &items[3];
items[4].down = &items[5];
items[4].child = 0;
items[4].parent = 0;
items[4].id = 5;
items[4].name = "Menu_5";
items[4].updateScreen_up = p_print_rows_on_oled_if_up;
items[4].updateScreen_down = p_print_rows_on_oled_if_down;
items[4].makeAction = 0;

items[5].up = &items[4];
items[5].down = &items[6];
items[5].child = 0;
items[5].parent = 0;
items[5].id = 6;
items[5].name = "Menu_6";
items[5].updateScreen_up = p_print_rows_on_oled_if_up;
items[5].updateScreen_down = p_print_rows_on_oled_if_down;
items[5].makeAction  = 0;

items[6].up = &items[5];
items[6].down = 0;
items[6].child = 0;
items[6].parent = 0;
items[6].id = 7;
items[6].name = "Menu_7";
items[6].updateScreen_up = p_print_rows_on_oled_if_up;
items[6].updateScreen_down = p_print_rows_on_oled_if_down;
items[6].makeAction  = 0;

////////////////////////////////////////////////////////////////////
// Creating next menu
items_menu_1[0].up = 0;
items_menu_1[0].down = &items_menu_1[1];
items_menu_1[0].id = 1;
items_menu_1[0].child = 0;
items_menu_1[0].parent = &items[0];
```

```
items_menu_1[0].name = "set par 1";
items_menu_1[0].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_1[0].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_1[0].makeAction = p_items_menu_1_set_par_1;

items_menu_1[1].up = &items_menu_1[0];
items_menu_1[1].down = &items_menu_1[2];
items_menu_1[1].child = 0;
items_menu_1[1].parent = &items[0];
items_menu_1[1].id = 2;
items_menu_1[1].name = "set par 2";
items_menu_1[1].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_1[1].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_1[1].makeAction = p_items_menu_1_set_par_2;

items_menu_1[2].up = &items_menu_1[1];
items_menu_1[2].down = &items_menu_1[3];
items_menu_1[2].child = 0;
items_menu_1[2].parent = &items[0];
items_menu_1[2].id = 3;
items_menu_1[2].name = "set par 3";
items_menu_1[2].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_1[2].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_1[2].makeAction = p_items_menu_1_set_par_3;

items_menu_1[3].up = &items_menu_1[2];
items_menu_1[3].down = &items_menu_1[4];
items_menu_1[3].child = 0;
items_menu_1[3].parent = &items[0];
items_menu_1[3].id = 4;
items_menu_1[3].name = "Выбрать";     // Name of item
items_menu_1[3].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_1[3].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_1[3].makeAction = p_do_it_function_menu_1;

items_menu_1[4].up = &items_menu_1[3];
items_menu_1[4].down = 0;
items_menu_1[4].child = 0;
```

```
items_menu_1[4].parent = &items[0];
items_menu_1[4].id = 5;
items_menu_1[4].name = "Выход";      // Name of item
items_menu_1[4].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_1[4].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_1[4].makeAction = p_return_from_menu;

////////////////////////////////////////////////////////////////
// Creating next menu
items_menu_2[0].up = 0;
items_menu_2[0].down = &items_menu_2[1];
items_menu_2[0].child = 0;
items_menu_2[0].parent = &items[1];
items_menu_2[0].id = 1;
items_menu_2[0].name = "set par 1";     // Name of item
items_menu_2[0].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_2[0].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_2[0].makeAction = p_items_menu_2_set_par_1;

items_menu_2[1].up = &items_menu_2[0];
items_menu_2[1].down = &items_menu_2[2];
items_menu_2[1].child = 0;
items_menu_2[1].parent = &items[1];
items_menu_2[1].id = 2;
items_menu_2[1].name = "Выбрать";     // Name of item
items_menu_2[1].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_2[1].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_2[1].makeAction = p_do_it_function_menu_2;

items_menu_2[2].up = &items_menu_2[1];
items_menu_2[2].down = 0;
items_menu_2[2].child = 0;
items_menu_2[2].parent = &items[1];
items_menu_2[2].id = 3;
items_menu_2[2].name = "Выход";      // Name of item
items_menu_2[2].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_2[2].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_2[2].makeAction = p_return_from_menu;
```

```
//////////////////////////////////////////////////////////////////
// Creating next menu
items_menu_3[0].up = 0;
items_menu_3[0].down = &items_menu_3[1];
items_menu_3[0].child = 0;
items_menu_3[0].parent = &items[2];
items_menu_3[0].id = 1;
items_menu_3[0].name = "Выбрать";     // Name of item
items_menu_3[0].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_3[0].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_3[0].makeAction = p_do_it_function_menu_3;

items_menu_3[1].up = &items_menu_3[0];
items_menu_3[1].down = 0;
items_menu_3[1].child = 0;
items_menu_3[1].parent = &items[2];
items_menu_3[1].id = 2;
items_menu_3[1].name = "Выход";      // Name of item
items_menu_3[1].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_3[1].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_3[1].makeAction = p_return_from_menu;

//////////////////////////////////////////////////////////////////
// Creating next menu
items_menu_4[0].up = 0;
items_menu_4[0].down = &items_menu_4[1];
items_menu_4[0].child = 0;
items_menu_4[0].parent = &items[3];
items_menu_4[0].id = 1;
items_menu_4[0].name = "set par 1";     // Name of item
items_menu_4[0].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[0].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[0].makeAction = p_items_menu_4_set_par_1;

items_menu_4[1].up = &items_menu_4[0];
items_menu_4[1].down = &items_menu_4[2];
items_menu_4[1].child = 0;
```

```
items_menu_4[1].parent = &items[3];
items_menu_4[1].id = 2;
items_menu_4[1].name = "set par 2";     // Name of item
items_menu_4[1].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[1].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[1].makeAction = p_items_menu_4_set_par_2;

items_menu_4[2].up = &items_menu_4[1];
items_menu_4[2].down = &items_menu_4[3];
items_menu_4[2].child = 0;
items_menu_4[2].parent = &items[3];
items_menu_4[2].id = 3;
items_menu_4[2].name = "set par 3";     // Name of item
items_menu_4[2].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[2].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[2].makeAction = p_items_menu_4_set_par_3;

items_menu_4[3].up = &items_menu_4[2];
items_menu_4[3].down = &items_menu_4[4];
items_menu_4[3].child = 0;
items_menu_4[3].parent = &items[3];
items_menu_4[3].id = 4;
items_menu_4[3].name = "set par 4";     // Name of item
items_menu_4[3].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[3].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[3].makeAction = p_items_menu_4_set_par_4;

items_menu_4[4].up = &items_menu_4[3];
items_menu_4[4].down = &items_menu_4[5];
items_menu_4[4].child = 0;
items_menu_4[4].parent = &items[3];
items_menu_4[4].id = 5;
items_menu_4[4].name = "set par 5";     // Name of item
items_menu_4[4].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[4].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[4].makeAction = p_items_menu_4_set_par_5;

items_menu_4[5].up = &items_menu_4[4];
```

```
items_menu_4[5].down = &items_menu_4[6];
items_menu_4[5].child = 0;
items_menu_4[5].parent = &items[3];
items_menu_4[5].id = 6;
items_menu_4[5].name = "set par 6";     // Name of item
items_menu_4[5].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[5].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[5].makeAction = p_items_menu_4_set_par_6;

items_menu_4[6].up = &items_menu_4[5];
items_menu_4[6].down = &items_menu_4[7];
items_menu_4[6].child = 0;
items_menu_4[6].parent = &items[3];
items_menu_4[6].id = 7;
items_menu_4[6].name = "set par 7";     // Name of item
items_menu_4[6].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[6].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[6].makeAction = p_items_menu_4_set_par_7;

items_menu_4[7].up = &items_menu_4[6];
items_menu_4[7].down = &items_menu_4[8];
items_menu_4[7].child = 0;
items_menu_4[7].parent = &items[3];
items_menu_4[7].id = 8;
items_menu_4[7].name = "set par 8";     // Name of item
items_menu_4[7].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[7].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[7].makeAction = p_items_menu_4_set_par_8;

items_menu_4[8].up = &items_menu_4[7];
items_menu_4[8].down = &items_menu_4[9];
items_menu_4[8].child = 0;
items_menu_4[8].parent = &items[3];
items_menu_4[8].id = 9;
items_menu_4[8].name = "Выбрать";     // Name of item
items_menu_4[8].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[8].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[8].makeAction = p_do_it_function_menu_4;
```

```
items_menu_4[9].up = &items_menu_4[8];
items_menu_4[9].down = 0;
items_menu_4[9].child = 0;
items_menu_4[9].parent = &items[3];
items_menu_4[9].id = 10;
items_menu_4[9].name = "Выход";      // Name of item
items_menu_4[9].updateScreen_up = p_print_rows_on_oled_if_up;
items_menu_4[9].updateScreen_down = p_print_rows_on_oled_if_down;
items_menu_4[9].makeAction = p_return_from_menu;
}
// ------------------------------------------------------------------------
----------------
void up(void)
{
 if (currentItem->up)
 {
     currentItem = currentItem->up;
     if (currentItem->updateScreen_up )
     {
         currentItem->updateScreen_up();
     }
 }
}
// ------------------------------------------------------------------------
----------------
void down(void)
{
 if (currentItem->down)
 {
     currentItem = currentItem->down;
     if (currentItem->updateScreen_down )
     {
         currentItem->updateScreen_down();
     }
 }
}
```

```
// ---------------------------------------------------------------------------
----------------
void enter(void)
{
bool status = true;

if (currentItem->makeAction)
{
 currentItem->makeAction();
 status = false;
}

if((currentItem->child) && (status == true))
{
 currentItem = currentItem->child;
 action();
}
}
// ---------------------------------------------------------------------------
----------------
void menu(void)
{
Menu_Init();        // Init all structures

print_menu_init();       // Print start menu and scrolingbar

HAL_Delay(10);

while(1)
{
 if(button_processed_status == 1)   // If buttons was pressed
 {

  button_processed_status = 1;
  switch (button_status)
  {
   case BOTTON_UP:
    up();
```

```
    break;
   case BUTTON_ENTER:
    enter();
    break;
   case BUTTON_DOWN:
    down();
    break;
  }
  button_status = BOTTON_DOESENT_PRESS;
 }
}}
```

Файл Signal/signal.c

```
#include "main.h"
#include <stdio.h>
#include <stdlib.h>

#include <stm32f0xx_it.h>

int t1,t2;

const int table[125]={
240,255,270,285,300,314,328,342,
356,369,381,393,404,415,425,434,
443,450,457,463,468,472,476,478,
480,480,480,478,476,472,468,463,
457,450,443,434,425,415,404,393,
381,369,356,342,328,314,300,285,
270,255,240,225,210,195,180,166,
152,138,124,111,99,87,76,65,
55,46,37,30,23,17,12,8,
4,2,0,0,0,2,4,8,
12,17,23,30,37,46,55,65,
76,87,99,111,124,138,152,166,
180,195,210,225,240,
//
240,255,270,285,300,314,328,342,
356,369,381,393,404,415,425,434,
443,450,457,463,468,472,476,478
};

void ConfigureHSE(void)
{
 __IO uint32_t StartUpCounter = 0, HSEStatus = 0;

 /******************************************************* ******************************/
 /* В качестве источника системных часов используется HSE */
 /******************************************************* ******************************/
 /* Конфигурация SYSCLK, HCLK, PCLK ---------------------------------------*/
 /* Включить HSE */
 RCC->CR |= RCC_CR_HSEON;
 /* Подождать, пока HSE будет готов, и если истекло время ожидания, выйти */
 do
  {
  HSEStatus = RCC->CR & RCC_CR_HSERDY;
  StartUpCounter++;
  } while((HSEStatus == 0) && (StartUpCounter != 5000));

 if ((RCC->CR & RCC_CR_HSERDY) != RESET)
  {
  HSEStatus = 0x01;
  }
 else
 {
 HSEStatus = 0x00;
 }

 if (HSEStatus == 0x01)
  {

    /* Включить буфер предварительной выборки и установить задержку флэш-памяти */
```

```
  FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_ACR_LATENCY;

  /* HCLK = SYSCLK / 1 */
  RCC->CFGR |= RCC_CFGR_HPRE_DIV1;

  /* PCLK = HCLK / 1 */
  RCC->CFGR |= RCC_CFGR_PPRE_DIV1;


  /* Выберите HSE в качестве источника системных часов */
  // RCC->CFGR &= ~(RCC_CFGR_SW_PLL);
  // RCC->CFGR |= RCC_CFGR_SW_HSE;
  RCC->CFGR |= RCC_CFGR_SWS_PLL;
  RCC->CFGR |= RCC_CFGR_SW_PLL;

  /* Подождите, пока HSE не будет использоваться в качестве источника системных часов */
  // while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSE)
  // {
  // }
  } /*else{  If HSE fails to start-up, the application will have wrong clock
 configuration. User can add here some code to deal with this error
 }*/
}

int main(void) {
ConfigureHSE();//только для кварцевого синхро
RCC->AHBENR |= RCC_AHBENR_GPIOBEN;
RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;
GPIOA->MODER = (GPIOA->MODER & ~(GPIO_MODER_MODER7)) | (GPIO_MODER_MODER7_1);
GPIOA->MODER = (GPIOA->MODER & ~(GPIO_MODER_MODER6)) | (GPIO_MODER_MODER6_1);
GPIOB->MODER = (GPIOA->MODER & ~(GPIO_MODER_MODER1)) | (GPIO_MODER_MODER1_1);
GPIOA->AFR[0] |= 0x11000000;
GPIOB->AFR[0] |= 0x00000010;

TIM3->PSC = 0; //APBCLK=48 МГц
TIM3->ARR = 480;
TIM3->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1PE;
TIM3->CCMR1 |= TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1 | TIM_CCMR1_OC2PE;
TIM3->CCMR2 |= TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1 | TIM_CCMR2_OC4PE;

TIM3->CCER |= TIM_CCER_CC1E;
TIM3->CCER |= TIM_CCER_CC2E;
TIM3->CCER |= TIM_CCER_CC4E;
TIM3->BDTR |= TIM_BDTR_MOE;
TIM3->CR1 |= TIM_CR1_CMS_0 | TIM_CR1_CEN;
TIM3->EGR |= TIM_EGR_UG;
SysTick_Config(1200); // 40 kHz
__enable_irq ();
t1=0;
t2=0;
while (1)
{
__WFI();
}
}
//////////////////////////////////////////////////////////////////////////////
void SysTick_Handler(void)
{
t1++;
```

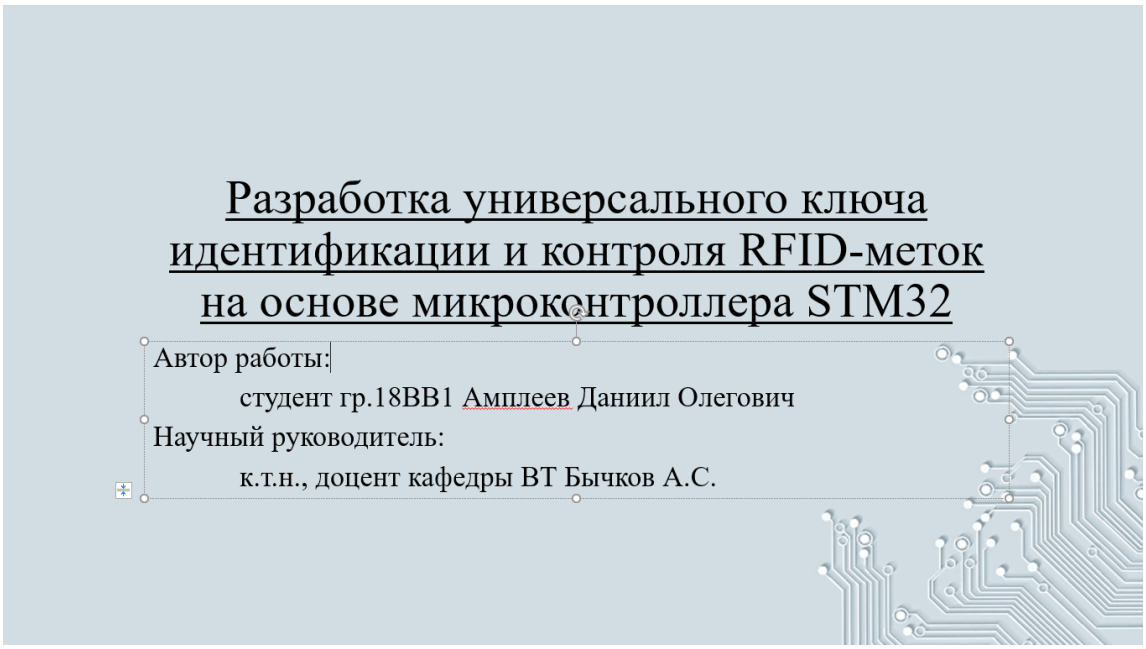
```
if(t1>=100){t1=0;}
t2++;
if(t2>=100){t2=0;}

TIM3->CCR1=(table[t2+25]);
TIM3->CCR2=(table[t1]);
TIM3->CCR4=table[t1+50];
}
```

Приложение В

(Обязательное)

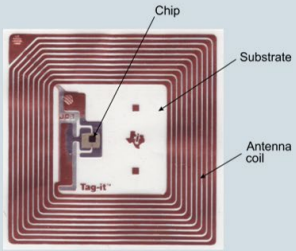
Универсальный ключа идентификации и контроля RFID-меток на основе микро-контроллера STM32

Презентация



Принципы работы RFID

-радиочастотная идентификация



RFID Tag

Среда разработки устройства

Mentor A Siemens Business

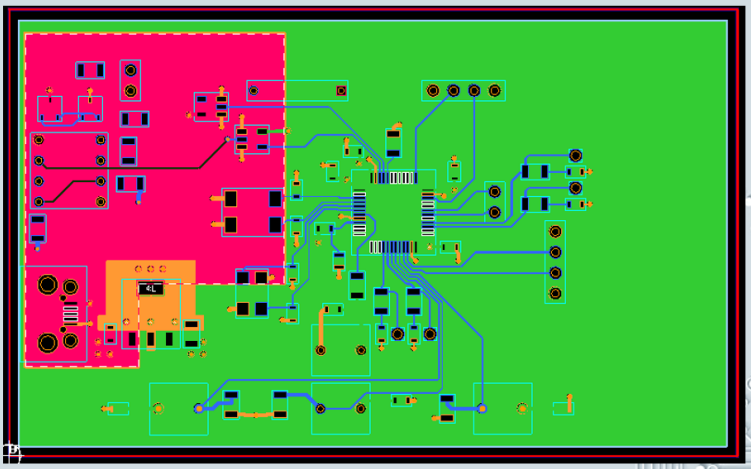
PADS®

ее основным и преимуществом перед другими САПР программами, является возможность децентрализованной работы нескольких инженеров над одним проектом, это достигается с помощью клиент-серверной системы.

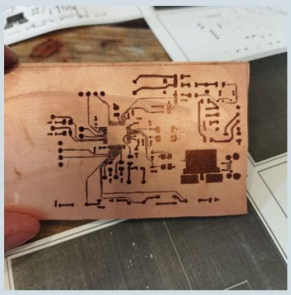
Среда разработки ПО

удобная среда разработки ПО для продуктов на базе STM32, обладает не только удобным интерфейсом но и широким функционалом. Основными преимуществами можно считать широкую базу программируемых микроконтроллеров и возможность задавать параметры основных параметров микроконтроллера напрямую из графического интерфейса

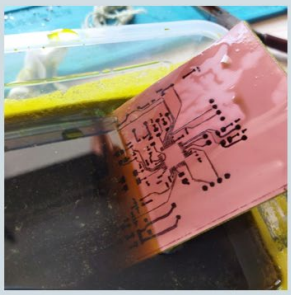
PADS Xpedition Layout



Произведенный продукт



Нанесение трафарета печатной платы на текстолит лазерноутюжным методом



Травление печатной платы в хлономжедлезе

Изготовление отверстий и населения припоя на сигнальных линий

Спасибо за внимание !